Intelligent Drivesystems, Worldwide Services

DE

# BU 0550

**PLC Logik**

für NORD Frequenzumrichter SK 54xE

NORD DRIVESYSTEMS

**N O R D Frequenzumrichter**

# Sicherheits- und Anwendungshinweise für Antriebsstromrichter

(gemäß: Niederspannungsrichtlinie 2006/95/EG)

**1. Allgemein**

Während des Betriebes können Antriebsstromrichter ihrer Schutzart entsprechend spannungsführende, blanke, gegebenenfalls auch bewegliche oder rotierende Teile, sowie heiße Oberflächen besitzen.

Bei unzulässigem Entfernen der erforderlichen Abdeckung, bei unsachgemäßem Einsatz, bei falscher Installation oder Bedienung, besteht die Gefahr von schweren Personen- oder Sachschäden.

Weitere Informationen sind der Dokumentation zu entnehmen.

Alle Arbeiten zum Transport, zur Installation und Inbetriebnahme sowie zur Instandhaltung sind **von qualifiziertem Fachpersonal** auszuführen (IEC 364 bzw. CENELEC HD 384 oder DIN VDE 0100 und IEC 664 oder DIN VDE 0110 und nationale Unfallverhütungsvorschriften beachten).

Qualifiziertes Fachpersonal im Sinne dieser grundsätzlichen Sicherheitshinweise sind Personen, die mit Aufstellung, Montage, Inbetriebsetzung und Betrieb des Produktes vertraut sind und über die ihrer Tätigkeit entsprechenden Qualifikationen verfügen.

**2. Bestimmungsgemäße Verwendung in Europa**

Antriebsstromrichter sind Komponenten, die zum Einbau in elektrische Anlagen oder Maschinen bestimmt sind.

Bei Einbau in Maschinen ist die Inbetriebnahme der Antriebsstromrichter (d.h. die Aufnahme des bestimmungsgemäßen Betriebes) solange untersagt, bis festgestellt wurde, dass die Maschine den Bestimmungen der EG-Richtlinie 2006/42EG (Maschinenrichtlinie) entspricht; EN 60204 ist zu beachten.

Die Inbetriebnahme (d.h. die Aufnahme des bestimmungsgemäßen Betriebes) ist nur bei Einhaltung der EMV-Richtlinie (2004/108/EG) erlaubt.

CE- gekennzeichnete Antriebsstromrichter erfüllen die Anforderungen der Niederspannungsrichtlinie 2006/95/EG. Es werden die in der Konformitätserklärung genannten harmonisierten Normen für die Antriebsstromrichter angewendet.

Die technischen Daten sowie die Angaben zu Anschlussbedingungen sind dem Leistungsschild und der Dokumentation zu entnehmen und unbedingt einzuhalten.

Die Antriebsstromrichter dürfen nur Sicherheitsfunktionen übernehmen, die beschrieben und ausdrücklich zugelassen sind.

**3. Transport, Einlagerung**

Die Hinweise für Transport, Lagerung und sachgemäße Handhabung sind zu beachten.

**4. Aufstellung**

Die Aufstellung und Kühlung der Geräte muss entsprechend den Vorschriften der zugehörigen Dokumentation erfolgen.

Die Antriebsstromrichter sind vor unzulässiger Beanspruchung zu schützen. Insbesondere dürfen bei Transport und Handhabung keine Bauelemente verbogen und/oder Isolationsabstände verändert werden. Die Berührung elektronischer Bauelemente und Kontakte ist zu vermeiden.

Antriebsstromrichter enthalten elektrostatisch gefährdete Bauelemente, die leicht durch unsachgemäße Behandlung beschädigt werden können. Elektrische Komponenten dürfen nicht mechanisch beschädigt oder zerstört werden (unter Umständen Gesundheitsgefährdung!).

**5. Elektrischer Anschluss**

Bei Arbeiten an unter Spannung stehenden Antriebsstromrichtern sind die geltenden nationalen Unfallverhütungsvorschriften (z.B. BGV A3, vorherige VBG 4) zu beachten.

Die elektrische Installation ist nach den einschlägigen Vorschriften durchzuführen (z.B. Leitungsquerschnitte, Absicherungen, Schutzleiteranbindung). Darüber hinausgehende Hinweise sind in der Dokumentation enthalten.

Hinweise für die EMV-gerechte Installation - wie Schirmung, Erdung, Anordnung von Filtern und Verlegung der Leitungen - befinden sich in der Dokumentation der Antriebsstromrichter. Diese Hinweise sind auch bei CE-gekennzeichneten Antriebsstromrichtern stets zu beachten. Die Einhaltung der durch die EMV-Gesetzgebung geforderten Grenzwerte liegt in der Verantwortung des Herstellers der Anlage oder Maschine.

**6. Betrieb**

Anlagen, in die Antriebsstromrichter eingebaut sind, müssen ggf. mit zusätzlichen Überwachungs- und Schutzeinrichtungen gemäß den jeweils gültigen Sicherheitsbestimmungen, z.B. Gesetz über technische Arbeitsmittel, Unfallverhütungsvorschriften usw. ausgerüstet werden.

Die Parametrierung und Konfiguration des Antriebsstromrichters ist so zu wählen, dass hieraus keine Gefahren entstehen.

Während des Betriebes sind alle Abdeckungen geschlossen zu halten.

**7. Wartung und Instandhaltung**

Nach dem Trennen der Antriebsstromrichter von der Versorgungsspannung dürfen spannungsführende Geräteteile und Leistungsanschlüsse wegen möglicherweise aufgeladener Kondensatoren nicht sofort berührt werden. Hierzu sind die entsprechenden Hinweisschilder auf dem Antriebsstromrichter zu beachten.

Weitere Informationen sind der Dokumentation zu entnehmen.

## Diese Sicherheitshinweise sind aufzubewahren!

## Dokumentation

Bezeichnung: BU 0550 DE
Mat. Nr.: 607 55 01
Gerätereihe: SK 540E, SK 545E

## Versionsliste

| Bezeichnung bisheriger Ausgaben | Software Version | Bemerkung |
|---|---|---|
| BU 0550 DE, September 2011<br>Mat. Nr. 607 5501 / 3911 | V 2.0 R0 | Erste Ausgabe |
| BU 0550 DE, Oktober 2011<br>Mat. Nr. 607 5501 / 4211 | V 2.0 R1 | Fehlerkorrektur / Funktionsänderung der Operatoren ANDN / ORN / XORN |
| BU 0550 DE, Februar 2013<br>Mat. Nr. 607 5501 / 0813 | V 2.1 R0 | Nachfolgende PLC Bausteine sind neu hinzugekommen: MC_Control, FB_Capture, FB_Read, FB_Write, FB_Weigh<br>Für einige MC wurde der BUSY Ausgang nachgepflegt.<br>Die Prozesswerte 35 bis 36, 62 bis 66 und 141 bis 147 sind neu aufgenommen worden.<br>Funktionelle Erweiterung im Kapitel „Sprungmarken“<br>Befehl „SQR“ in „SQRT“ umbenannt<br>der Eingang **RAMONLY** ist neu im MC_WriteParameter16/32<br>der Eingang **MODE** ist neu im MC_Home<br>Namen für die Prozesswerte korregiert<br>Anzahl der PLC Fehler auf 16 erhöht<br>Adressierungsmöglichkeiten für PDO verbessert<br>Beispielprogramm ergänzt |
| | | |
| | | |

## Herausgeber

**Getriebebau NORD GmbH & Co. KG**
Rudolf-Diesel-Str. 1 • D-22941 Bargteheide • http://www.nord.com/
Telefon +49 (0) 45 32 / 289-0 • Fax +49 (0) 45 32 / 289-2555

**HINWEIS**

Diese Zusatzbetriebsanleitung ist nur in Verbindung mit der Betriebsanleitung des jeweiligen Frequenzumrichters *(Handbuch BU0500)* gültig.

## Bestimmungsgemäße Verwendung der Frequenzumrichter

Die **Einhaltung** der Betriebsanleitung ist die **Voraussetzung für störungsfreien Betrieb** und die Erfüllung eventueller Garantieansprüche. **Lesen Sie deshalb zuerst die Betriebsanleitung** bevor Sie mit dem Gerät arbeiten!

Die Betriebsanleitung enthält **wichtige Hinweise zum Service.** Sie ist deshalb in **der Nähe des Gerätes** aufzubewahren.

Die NORD Frequenzumrichter sind Geräte für industrielle und gewerbliche Anlagen zum Betreiben von Drehstrom-Asynchronmotoren mit Kurzschlussläufer und **P**ermanent **M**agnet **S**ynchron **M**otoren – PMSM *(ab SK 54xE)*. Diese Motoren müssen zum Betrieb an Frequenzumrichtern geeignet sein, andere Lasten dürfen nicht an die Geräte angeschlossen werden.

Die Frequenzumrichter SK 5xxE sind Geräte für den stationären Aufbau in Schaltschränken. Alle Angaben zu den technischen Daten und den zulässigen Bedingungen am Einsatzort sind unbedingt einzuhalten.

Die Inbetriebnahme (Aufnahme des bestimmungsgemäßen Betriebs) ist so lange untersagt, bis festgestellt ist, dass die Maschine die EMV-Richtlinie 2004/108/EG einhält und die Konformität des Endproduktes beispielsweise mit der Maschinenrichtlinie 2006/42/EG feststeht (EN 60204 beachten).

© Getriebebau NORD GmbH & Co. KG, 2013

**1 ALLGEMEINES ..... 9**
1.1 Spezifikation der PLC ..... 9
1.2 PLC Aufbau ..... 9
1.2.1 Speicher ..... 9
1.2.2 Prozessabbild ..... 10
1.2.3 Programm Task ..... 10
1.2.4 Sollwert Verarbeitung ..... 11
1.2.5 Datenverarbeitung über Akku ..... 11
1.3 Funktionsumfang ..... 12
1.3.1 Motion Control Lib ..... 12
1.3.2 Elektronisches Getriebe mit Fliegender Säge ..... 12
1.3.3 Visualisierung ..... 12
1.3.4 Prozessregler ..... 13
1.3.5 CANopen Kommunikation ..... 13
**2 ERSTELLEN VON PLC PROGRAMMEN ..... 14**
2.1 Laden, Speichern & Drucken ..... 14
2.2 Editor ..... 14
2.2.1 Variablen und FB Deklaration ..... 15
2.2.2 Eingabefenster ..... 15
2.2.3 Watch- & Breakpoint Anzeigefenster ..... 16
2.2.4 PLC Meldungsfenster ..... 16
2.3 Programm in den FU laden ..... 16
2.4 Debugging ..... 17
2.4.1 Beobachtungspunkte (Watchpoints) ..... 17
2.4.2 Haltepunkte (Breakpoints) ..... 17
2.4.3 Einzelschritt (Single Step) ..... 17
2.5 PLC Konfiguration ..... 18
2.6 PLC Programmstart ..... 18
2.7 Programmbeispiel ..... 19
**3 AWL (INSTRUCTION LIST, IL) ..... 20**
3.1 Allgemein ..... 20
3.1.1 Datentypen ..... 20
3.1.2 Literale ..... 20
3.1.3 Kommentare ..... 21
3.1.4 Sprungmarke ..... 21
3.1.5 Funktionsaufrufe ..... 22
3.1.6 Bitweiser Zugriff auf Variablen ..... 22
3.2 Operatoren ..... 22
3.2.1 Lade- und Speicheroperatoren ..... 22
3.2.2 Arithmetische Operatoren ..... 24
3.2.3 Erweiterte mathematische Operatoren ..... 29
3.2.4 Bit Operationen ..... 31
3.2.5 Vergleichs Operatoren ..... 37
3.3 Sprünge ..... 40
3.3.1 JMP ..... 40
3.3.2 JMPC ..... 40
3.3.3 JMPCN ..... 40
3.4 Typkonvertierung ..... 41
3.4.1 BYTE_TO_BOOL ..... 41
3.4.2 BOOL_TO_BYTE ..... 41
3.4.3 INT_TO_BYTE ..... 41
3.4.4 BYTE_TO_INT ..... 42
3.4.5 DINT_TO_INT ..... 42
3.4.6 INT_TO_DINT ..... 42

3.5 Funktionsblöcke ..... 43
3.5.1 Standard ..... 43
3.5.2 Motion Control ..... 50
3.5.3 Elektronisches Getriebe mit „fliegender Säge“ ..... 60
3.5.4 FB_FunctionCurve ..... 63
3.5.5 FB_PIDT1 ..... 64
3.5.6 Visualisierung ParameterBox ..... 66
3.5.7 CANopen ..... 69
3.5.8 Erfassen schneller Ereignisse ( FB_Capture ) ..... 74
3.5.9 Zugriff auf Speicherbereiche des Frequenzumrichters ..... 75
3.5.10 Wiegefunktion (FB_Weigh) ..... 77

**4 DETAILS FREQUENZUMRICHTER ..... 78**
4.1 Parameter ..... 78
4.1.1 Betriebsanzeigen ..... 78
4.1.2 Reglungs- / PLC Parameter ..... 78
4.1.3 Steuerklemmen ..... 80
4.1.4 Zusatzparameter ..... 83
4.1.5 Informationsparameter ..... 85
4.2 Prozesswerte ..... 87
4.2.1 Ein- und Ausgänge ..... 87
4.2.2 Soll- und Istwerte PLC ..... 88
4.2.3 Bus Soll- und Istwerte ..... 89
4.2.4 ControlBox und ParameterBox ..... 90
4.2.5 Infoparameter ..... 90
4.2.6 PLC Fehler ..... 91
4.2.7 PLC Parameter ..... 92
4.3 PLC Störmeldungen ..... 93

**5 VERZEICHNISSE / REGISTER ..... 94**
5.1 Abkürzungen ..... 94
5.2 Abbildungen ..... 95
5.3 Tabellen ..... 95
5.4 Stichwort-Verzeichnis ..... 97

# 1 Allgemeines

Der NORD Frequenzumrichter SK54xE enthält eine Logikverarbeitung, welche an die für Speicherprogrammierbare Steuerungen (SPS / PLC) geltende Norm IEC61131-3 angelehnt ist. Die Reaktionsgeschwindigkeit oder Rechenleistung dieser PLC ist geeignet kleinere Aufgaben im Umfeld des Umrichters zu übernehmen. So können Umrichter-Eingänge oder über einen Feldbus ankommende Informationen überwacht, ausgewertet und in entsprechende Sollwerte für den Frequenzumrichter weiterverarbeitet werden. Im Zusammengehen mit anderen NORD Geräten ist auch eine Visualisierung von Anlagenzuständen und Eingabe von speziellen Kundenparametern möglich. Somit ergibt sich im begrenzten Bereich ein Einsparungspotential über das Weglassen einer bisherigen externen PLC Lösung.

Als Programmiersprache wird AWL unterstützt. AWL ist eine maschinennahe textbasierende Programmiersprache, deren Umfang und Anwendung in der IEC61131-3 festgelegt ist.

**HINWEIS**

Die Programmierung und der Download in den Frequenzumrichter erfolgt ausschließlich über die NORD Software NORDCON.

## 1.1 Spezifikation der PLC

| Funktion | Spezifikation |
|---|---|
| Standard | An IEC61131-3 angelehnt |
| Sprache | Instruction List ( IL ) |
| Task | Ein zyklischer Task, Programmaufruf alle 5 ms |
| Rechenleistung | Zirka 200 AWL Befehle auf 1ms |
| Programmspeicher | 4032 Byte für Merker, Funktionen und das PLC Programm |
| Max. mögliche Anzahl von Befehlen | ungefähr 1280 Befehle<br>Achtung! Dies ist ein Durchschnittswert, eine starke Verwendung von Merkern, Prozessdaten und Funktionen minimiert die möglich Zeilenanzahl erheblich, siehe Abschnitt Ressourcen |
| Frei ansprechbare CAN Mailboxen | 20 |

Tabelle 1 Spezifikation der PLC

## 1.2 PLC Aufbau

### 1.2.1 Speicher

Der Speicher in der PLC wird in Programm- und Merkerspeicher unterteilt. Im Bereich des Merkerspeichers werden neben den Variablen auch die Instanzen von Funktionsblöcken abgelegt. Eine Instanz ist ein Speicherbereich, in dem alle internen Ein- und Ausgabevariablen eines FB abgelegt werden. Jede FB Deklaration benötig eine eigene Instanz. Die Grenze zwischen Programm- und Merkerspeicher wird dynamisch festgelegt, abhängig von der Größe des Merkerbereiches.

Abbildung 1 Speicheraufbau

Im Merkerspeicher werden im Bereich Variablen zwei verschiedene Klassen abgelegt:

[VAR]
Speichervariable zum Ablegen von Hilfsinformationen und Zuständen. Variablen dieses Typs werden bei jedem Start der PLC neu initialisiert. Während des zyklischen Ablaufs der PLC bleiben die Speicherinhalte erhalten.

[VAR_ACCESS]

Dient zum Einlesen und Beschreiben von Prozessdaten (Eingänge, Ausgänge, Sollwerte, usw.) des Frequenzumrichters. Diese Werte werden bei jedem PLC Zyklus neu erzeugt.

### 1.2.2 Prozessabbild

Der Umrichter verfügt über etliche physikalische Größen wie Drehmoment, Drehzahl, Position, Eingänge, Ausgänge, usw. Diese Größen unterteilen sich in Ist- und Sollwerte. Sie können im Prozessabbild der PLC geladen und beeinflusst werden. Die benötigten Prozesswerte müssen in der Variablenliste unter der Klasse VAR_ACCESS definiert werden.

Mit jedem PLC Zyklus werden alle in der Variablenliste definierten Prozessdaten des Umrichters neu eingelesen. Am Ende jedes PLC Zyklus werden die beschreibbaren Prozessdaten wieder dem Umrichter übergeben, siehe nachfolgende Abbildung.

Abbildung 2 Prozessabbild

Aufgrund dieses Ablaufes ist es wichtig, einen zyklischen Programmablauf zu programmieren. Das Programmieren von Schleifen, um auf bestimmte Ereignisse zu warten (z.B. Pegeländerung an einem Eingang), führt nicht zum gewünschten Ergebnis.

Bei Funktionsblöcken, die auf Prozesswerte zugreifen, ist dieses Verhalten anders. Hier werden die Prozesswert mit dem Aufruf des Funktionsblockes gelesen und bei Beendigung des Blockes werden die Prozesswerte sofort geschrieben.

**HINWEIS**

Werden Motion Blöcke MC_Power, MC_Reset, MC_MoveVelocity, MC_Move…, MC_Home oder MC_Stop verwendet, dann dürfen die Prozesswerte „PLC_Control_Word“ und „PLC_Set_Val1“ bis „PLC_Set_Val5“ nicht verwendet werden. Da sonst die Werte in der Variablenliste immer die Änderung des Funktionsblockes überschreiben würden.

### 1.2.3 Programm Task

Die Programmausführung in der PLC erfolgt in einer einzigen Task. Die Task wird zyklisch alle 5ms aufgerufen und ihre max. Bearbeitungsdauer beträgt 3ms. Kann ein längeres Programm in dieser Zeit nicht abgearbeitet werden, dann wird die Programmausführung unterbrochen und in der nächsten 5ms Task fortgeführt.

### 1.2.4 Sollwert Verarbeitung

Der Umrichter verfügt über eine Vielzahl von Sollwertquellen, die letztendlich über mehrere Parameter zu einem resultierenden Frequenzumrichter Sollwert miteinander verknüpft werden.

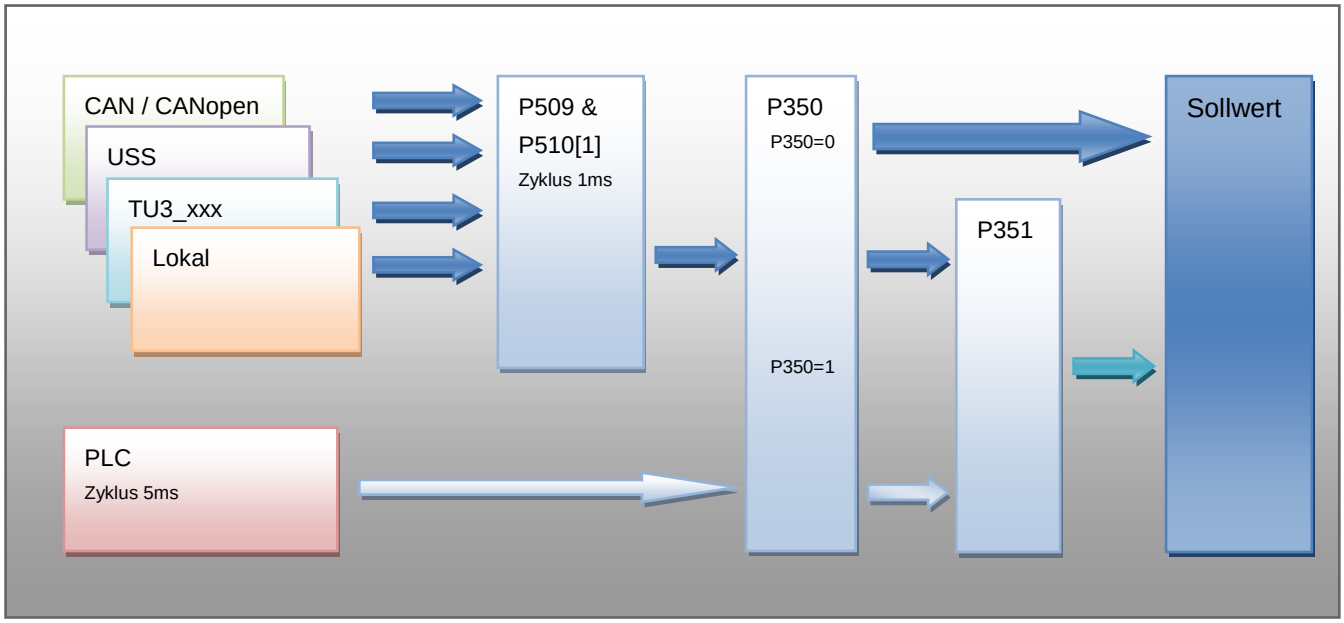



Abbildung 3 Erzeugung von Steuerwort & Hauptsollwerten aus der Vielzahl möglicher Quellen

Bei aktivierter PLC (P350=1) erfolgt über die P509 & P510[-01] eine Vorselektion der von außen eingehenden Sollwerte (Hauptsollwerte). Über den P351 wird dann letztlich entschieden, welche Sollwerte von der PLC oder den über P509/P510[-01] eingehenden Werten genommen wird. Auch ein Mix aus beiden ist möglich.

Bei den Nebensollwerten (P510[-02]) verändert sich im Zusammenhang mit der PLC Funktion nichts. Alle Nebensollwertquellen und die PLC übergeben ihre Nebensollwerte gleichberechtigt an den Frequenzumrichter.

### 1.2.5 Datenverarbeitung über Akku

Der Akkumulator bildet die zentrale Recheneinheit der PLC. Fast alle AWL-Befehle funktionieren nur im Zusammenhang mit dem Akkumulator. In der NORD PLC existieren gleich drei Akkumulatoren. Dabei handelt es sich um die 32Bit großen Akku1 und Akku2, sowie das AE im Format BOOL.

Das AE wird für alle booleschen Lade-, Speicher- und Vergleichsoperationen herangezogen. Wird ein boolescher Wert geladen so wird er im AE dargestellt. Vergleichsoperanden liefern das Ergebnis im AE ab und bedingte Sprünge werden aufgrund des AE ausgelöst.

Akku1 und Akku2 werden für alle Operanden im Datenformat BYTE, INT und DINT verwendet. Bei Akku1 handelt es sich um den Hauptakkumulator während Akku2 nur Hilfsfunktionen übernimmt. Alle Lade und Speicheroperanden laufen über Akku1. Alle arithmetischen Operatoren speichern ihr Ergebnis unter Akku1 ab. In Akku2 wird bei jedem Ladebefehl der Inhalt von Akku1 verschoben. Ein nachfolgender Operator kann dann beide Akkumulatoren miteinander verknüpfen oder auswerten und das Ergebnis wieder in Akku1, der im Folgenden auch allgemein als „Akku“ bezeichnet wird. speichern.

## 1.3 Funktionsumfang

Die PLC unterstützt eine Vielzahl von Operatoren, Funktionen und Standardfunktionsbausteinen, die in der IEC1131-3 definiert sind. Eine detaillierte Darstellung ist in den nachfolgenden Kapiteln (Kap. 3 und 4 ) enthalten.

Des Weiteren werden Funktionsblöcke erläutert (siehe Kap. 3.5 ), die zusätzlich unterstützt werden.

### 1.3.1 Motion Control Lib

Die Motion Control Lib ist an die PLCopen Spezifikation „Function blocks for motion control“ angelehnt. In ihr sind hauptsächlich Funktionsblöcke zum Verfahren des Antriebs enthalten. Zusätzlich werden auch Funktionsblöcke zum Lesen und Schreiben von FU Parametern bereitgestellt.

### 1.3.2 Elektronisches Getriebe mit Fliegender Säge

Der Frequenzumrichter verfügt über die Funktionen *elektronisches Getriebe* (Gleichlauf im Positioniermodus) und *Fliegende Säge*. Über diese Funktionen kann der Umrichter mit einem anderen Antrieb winkelsynchron mitfahren. Weiterhin ist es über die Zusatzfunktion *Fliegende Säge* möglich, sich positionsgenau auf einen fahrenden Antrieb zu synchronisieren. Der Betriebsmodus *elektronisches Getriebe* kann jederzeit gestartet und beendet werden. Damit ist eine Kombination von klassischer Lageregelung mit ihren Verfahrbefehlen und Getriebefunktion möglich.

Für die Getriebefunktion wird an der Masterachse zwingend ein NORD Frequenzumrichter mit internem CAN-Bus benötigt.

### 1.3.3 Visualisierung

Mit Hilfe einer ControlBox bzw. einer ParameterBox sind die Visualisierung des Betriebszustandes und die Parametrierung des Frequenzumrichters möglich.

Alternativ können auch über die CANopen Master Funktionalität der PLC CAN-Bus Panels zur Anzeige von Informationen verwendet werden.

#### 1.3.3.1 ControlBox

Die einfachste Variante zur Visualisierung ist die ControlBox. Über zwei Prozesswerte kann auf das 4 stellige Display und den Zustand der Tastatur zugegriffen werden. Damit können sehr schnell einfache HMI Applikationen erstellt werden.

Damit die PLC auf die Anzeige zugreifen kann muss der P001 auf „PLC-Ctrlbox Wert“ eingestellt werden. Eine weitere Besonderheit ist, dass das Parametermenü nicht mehr über die Pfeiltasten erreicht wird. Stattdessen müssen die „On“ und „Enter“ Taste zeitgleich betätigt werden.

#### 1.3.3.2 ParameterBox

Im Visualisierungsmodus kann über die PLC jedes der 80 Zeichen im P-Box Display (4 Zeilen a 20 Zeichen) gesetzt werden. Es ist möglich Zahlen wie auch Texte zu übertragen. Weiterhin können Tastatureingaben auf der P-Box von der PLC erfasst werden. Damit ist eine Realisierung komplexerer HMI Funktionen ( Anzeige von Istwerten, Bildwechsel, Übergabe von Sollwerten, usw. ) möglich. Der Zugriff auf die P-Box Anzeige erfolgt über Funktionsblöcke in der PLC.

Die Visualisierung erfolgt über die Betriebswertanzeige der ParameterBox. Der Inhalt der Betriebswertanzeige wird über den P-Box Parameter P1003 eingestellt. Dieser Parameter befindet sich unter dem Hauptmenüpunkt „Anzeige“. P1003 muss auf den Wert „PLC-Anzeige“ eingestellt werden. Über die Pfeiltasten Rechts oder Links kann die Betriebswertanzeige danach wieder angewählt werden. Hier wird jetzt das von der PLC kontrollierte Display angezeigt. Diese Einstellung bleibt auch nach einem erneuten Einschalten erhalten.

### 1.3.4 Prozessregler

Der Prozessregler ist ein PID-T1 – Regler mit begrenzter Ausgangsgröße. Mit Hilfe dieses Funktionsbausteines können in der PLC auf einfache Weise komplexe Regelungen aufgebaut werden, über die sich etliche Prozesse, wie z.B. Druckregelungen, deutlich eleganter lösen lassen als mit den häufig verwendeten Zweipunktreglern.

### 1.3.5 CANopen Kommunikation

Neben den standardmäßig vorhandenen Kommunikationskanälen bietet die PLC noch weitere Möglichkeiten zu kommunizieren. Über den internen CAN Bus des Umrichters (Anschluss über die RJ45 Buchsen) kann dieser mit anderen Geräten zusätzliche Kommunikationsbeziehungen aufbauen. Das dabei verwendete Protokoll ist CANopen. Die Kommunikationsbeziehungen sind dabei auf den PDO Datentransfer und NMT Kommandos beschränkt. Die per Standard im Frequenzumrichter vorhandene CANopen Kommunikation über SDO, PDO1, PDO2 und Broadcast bleibt von dieser PLC - Funktion unbeeinträchtigt.

**PDO (Prozess Daten Objects)**

Über PDO können andere Frequenzumrichter gesteuert und überwacht werden. Es ist aber auch möglich Geräte anderer Anbieter an die PLC anzubinden. Dies können IO-Baugruppen, CANopen Geber, Panels, usw. sein. Damit kann die Anzahl der Ein/Ausgänge des Frequenzumrichters beliebig erweitert werden, auch analoge Ausgänge wären dann möglich.

**NMT (Network Management Objects)**

Alle CANopen Geräte müssen vom Busmaster in den CANopen Bus State „Operational“ gebracht werden. Erst in diesem Buszustand ist eine PDO Kommunikation möglich. Wenn sich kein Busmaster in dem CANopen Bus befindet, muss dies durch die PLC erfolgen. Für diesen Zweck gibt es den Funktionsbaustein FB_NMT.

# 2 Erstellen von PLC Programmen

Die Erstellung der PLC Programme erfolgt ausschließlich über das PC-Programm **NORDCON**. Der PLC Editor wird entweder über den Menüpunkt „Datei/Neu/PLC Programm“ oder durch das Symbol geöffnet. Diese Schaltfläche ist nur aktiv, wenn in der Geräteübersicht ein Frequenzumrichter mit PLC Funktionalität den Fokus hat.

## 2.1 Laden, Speichern & Drucken

Die Funktionen Laden, Speichern und Drucken erfolgen über die entsprechenden Einträge im Hauptmenü oder die Symbolleisten.

Beim Öffnen ist es empfehlenswert im Windowsdialog „Öffnen“ den Dateityp auf „PLC Programm“ (*.awl / *.awlx) zu setzen. Damit werden nur noch Dateien, die vom PLC Editor gelesen werden können, angezeigt.

Soll das erstellte PLC Programm gespeichert werden, dann muss das Windowsfester vom PLC Editor aktiv sein. Das PLC Programm wird durch Betätigen von „Speichern“ oder „Speichern unter“ gesichert. Bei der Operation „Speichern unter“ kann dies auch am Eintrag des Dateityp (540E PLC (*.awlx)) erkannt werden.

Für das Drucken des PLC Programmes muss auch das entsprechende PLC Fenster aktiv sein. Der Ausdruck wird dann über „Datei/Drucken“ oder das passende Symbol gestartet.

## 2.2 Editor

Der PLC – Editor ist in vier verschiedene Fenster aufgeteilt.

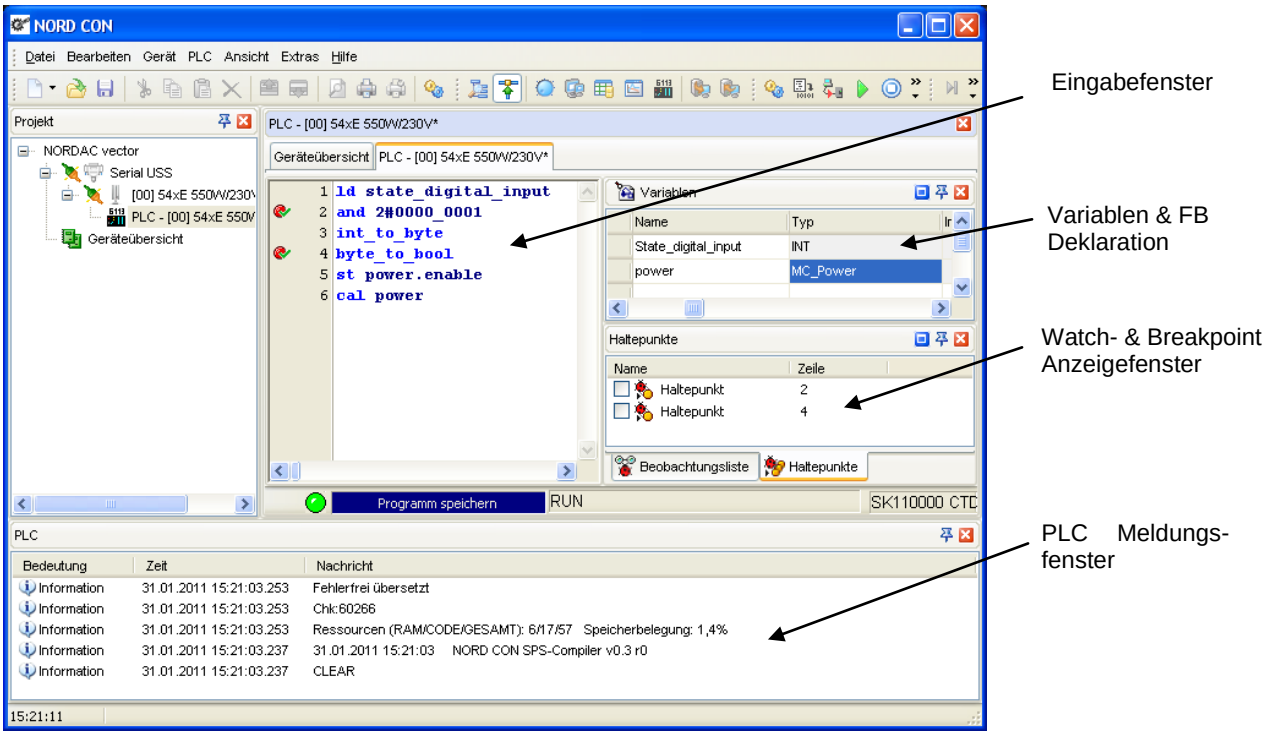



Abbildung 4 PLC Editor

Die einzelnen Fenster werden in den nachfolgenden Abschnitten näher erläutert.

### 2.2.1 Variablen und FB Deklaration

In diesem Fenster werden alle im Programm benötigten Variablen, Prozesswerte und Funktionsblöcke deklariert.

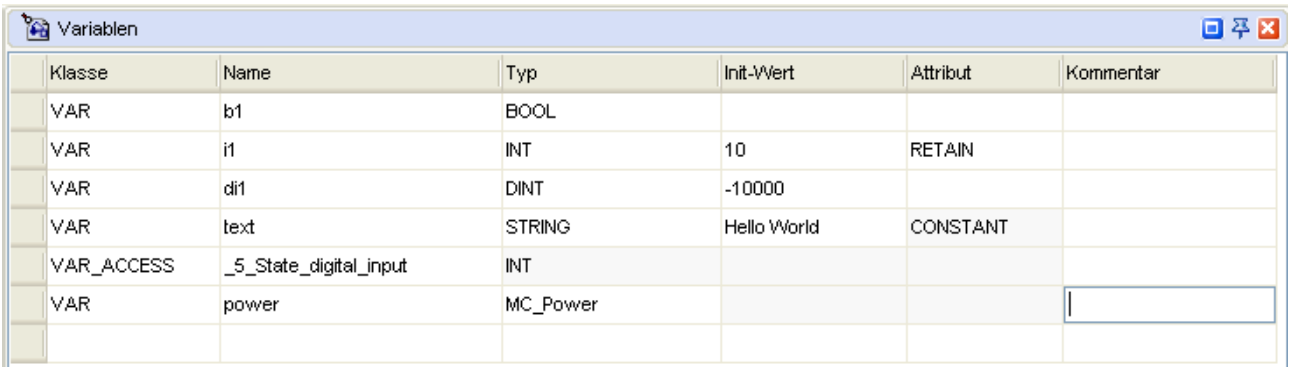



Abbildung 5 Deklarationsfenster

**Variablen**
Variablen werden angelegt, indem die *Klasse* „VAR“ eingestellt wird. Der *Name* für die Variable ist frei wählbar. Im Feld *Typ* kann zwischen BOOL, BYTE, INT und DINT gewählt werden. Für die Variablen kann eine Startinitialisierung unter *Init-Wert* eingetragen werden.

**Prozesswerte**
Diese werden angelegt indem unter *Klasse* der Eintrag „VAR_ACCESS“ selektiert wird. Der *Name* ist nicht frei wählbar und das Feld *Init-Wert* ist für diesen Typ gesperrt.

**Funktionsbausteine**
Unter *Klasse* wird der Eintrag „VAR“ selektiert. Der *Name* für die jeweilige Instanz des Funktionsbausteins (FB) ist frei wählbar. Der gewünschte FB wird unter *Typ* selektiert. Ein *Init-Wert* ist für FB nicht einstellbar.

Alle Menüpunkte, die das Variablenfenster betreffen, werden über das Kontextmenü (rechte Maustaste) aufgerufen. Hierüber können Einträge hinzugefügt und gelöscht werden. Sowie Variablen und Prozessvariablen zur Beobachtung (Watchpoint Funktion) oder zum Debuggen (Breakpoint) aktiviert werden.

### 2.2.2 Eingabefenster

Das Eingabefenster dient zur Programmeingabe und auch Darstellung des AWL-Programmes. Es verfügt über folgende Funktionen:

- Syntax Hervorhebung
- Lesezeichen
- Variablen Deklaration
- Debugging

**Syntax Hervorhebung**
Werden der Befehl und die ihm zugeordnete Variable vom Editor erkannt, dann werden der Befehl blau und die Variable schwarz dargestellt. Solange dies nicht der Fall ist, erfolgt die Darstellung in dünner, schräger, schwarzer Schrift.

**Lesezeichen**
Da Programme im Editor durchaus eine beträchtliche Länge erreichen können, ist es möglich über die Funktion *Lesezeichen* wichtige Stellen im Programm zu markiert und gezielt anzuspringen. Zur Markierung einer Zeile muss sich der Cursor in der betreffenden Zeile befinden. Über den Menüpunkt „Lesezeichen umschalten“ (rechte Maustaste Menü) wird die Zeile mit dem gewünschten Lesezeichen markiert. Angesprungen werden die Lesezeichen über den Menüpunkt „Gehe zu Lesezeichen“.

**Variablen Deklaration**
Über das Editor Menü „Variable hinzufügen“ (rechte Maustaste) können vom Editor aus neue Variablen deklariert werden.

**Debugging**
Für die Funktion *Debugging* werden im Editor die Position der Break- und Watchpoints festgelegt. Dies kann über die Menüpunkte „Haltepunkt umschalten“ (Breakpoints) und „Beobachtungspunkt umschalten“ (Watchpoints) passieren. Die Position von Breakpoints kann zusätzlich über einen Klick auf der linken Randleise des Editorfensters festgelegt werden.

Variablen und Prozesswerte, die während des Debuggings aus dem Frequenzumrichter ausgelesen werden sollen, müssen markiert werden. Dies kann im Editor über die Menüpunkte „Variable debuggen“ und „Variable beobachten“ erfolgen. Dazu muss die entsprechende Variable markiert sein, bevor der gewünschte Menüpunkt angewählt wird.

### 2.2.3 Watch- & Breakpoint Anzeigefenster

Dieses Fenster verfügt über zwei Tab Reiter die nachfolgend erläutert werden.

**Haltepunkte**
In diesem Fenster sind alle gesetzten Breakpoint und Watchpoints zu sehen. Sie können über die Checkboxen ein-/ausgeschaltet und über die „Entfernen Taste“ gelöscht werden. Über die rechte Maustaste kann ein entsprechendes Menü aufgerufen werden.

**Beobachtungsliste**
Hier werden alle zur Beobachtung ausgewählten Variablen dargestellt. In der Spalte *Wert* wird ihr aktueller Inhalt dargestellt. Über die Spalte *Anzeige* kann das Darstellungsformat ausgewählt werden.

### 2.2.4 PLC Meldungsfenster

In diesem Fenster werden alle Status- und Fehlermeldungen der PLC eingetragen. Für ein korrekt übersetztes Programm erscheint die Meldung „Fehlerfrei übersetzt“. Eine Zeile tiefer wird der Ressourcenverbrauch angezeigt. Bei Fehlern im PLC Programm erscheint die Meldung „Fehler X“, in X wird die Anzahl der Fehler dargestellt. In den folgenden Zeilen erscheint die konkrete jeweilige Fehlermeldung im Format:

[ Zeilennummer]: Fehlerbeschreibung

## 2.3 Programm in den FU laden

Damit ein Programm in den Frequenzumrichter geladen werden kann, muss das PLC Fenster Online und das Programm fehlerfrei übersetzt sein.

Online ist das PLC Fenster, wenn das PLC Symbol im Baumdiagramm dem FU (54xE…) untergeordnet ist, siehe nachfolgendes Bild.

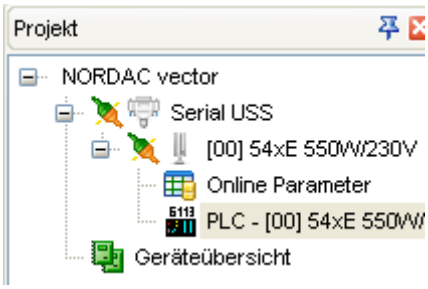



Abbildung 6 NORDCON Übersichtsfenster

Erreicht wird dies, wenn das PLC Fenster über das PLC Symbol geöffnet wird.

Wurde eine PLC Programm im Offline Fenster geöffnet, kann über den Menüpunkt "PLC -> Verbinden" dem PLC Programm ein Gerät zugewiesen werden. Mit dem Menüpunkt "PLC -> Trennen" kann man das PLC Programm wieder in den Offline Modus versetzen.

Über das Symbol wird das PLC Programm in den FU geladen und auch sofort gespeichert. Über das Symbol kann das Programm danach im Frequenzumrichter gestartet werden.

## 2.4 Debugging

Da Programme nur in seltenen Fällen auf Anhieb funktionieren bietet die NORD PLC einige Möglichkeiten zur Fehlerfindung. Diese Möglichkeiten lassen sich grob in zwei Punkte unterteilen, auf die jetzt nachfolgend eingegangen wird.

### 2.4.1 Beobachtungspunkte (Watchpoints)

Die einfachste Debugging Variante ist die Watchpoint Funktion. Sie bietet einen schnellen Überblick über das Verhalten einiger Variablen. Dazu wird an beliebiger Stelle im Programm ein Beobachtungspunkt gesetzt. Wenn die PLC diese Programmzeile abarbeitet, werden bis zu 5 Werte gespeichert und in der Beobachtungsliste angezeigt (Fenster „Haltepunkt“). Die 5 zu beobachtenden Werte können im Eingabefenster oder Variablenfenster über das Kontextmenü (rechte Maustaste) ausgewählt werden.

### 2.4.2 Haltepunkte (Breakpoints)

Über Haltepunkte ist es möglich das PLC Programm gezielt an einer gewünschten Programmzeile zu stoppen. Wenn die PLC in einen Haltepunkt hineinläuft werden das AE, Akku1 und Akku2 ausgelesen, sowie alle Variablen, die über den Menüpunkt „Variable debuggen“ (rechte Maus Menü) selektiert wurden.

Es können bis zu 5 Breakpoints im PLC Programm gesetzt werden.

Gestartet wird diese Funktion über das Symbol . Das Programm läuft nun solange bis ein Haltepunkt ausgelöst wird. Eine erneute Betätigung der Symbolleiste lässt das Programm wieder frei laufen bis der nächste Haltepunkt kommt.

Soll das Programm wieder frei laufen, so wird das Symbol betätigt.

### 2.4.3 Einzelschritt (Single Step)

Mit dieser Debugging Methode ist es möglich das PLC Programm Zeile für Zeile in Einzelschritten abzuarbeiten. Mit jedem Einzelschritt werden alle ausgewählten Variablen aus der FU PLC ausgelesen und im Fenster „Beobachtungsliste“ angezeigt. Die zu beobachtenden Werte können im Eingabefenster oder Variablenfenster über das rechte Maustastenmenü ausgewählt werden.

Voraussetzung für das Debugging in Einzelschritten ist, dass vor dem Start des Debugging mindestens ein Haltepunkt (siehe Kapitel 2.4.2 ) gesetzt wurde. Durch Betätigung des Symbols wird der Debugging Mode eingeschaltet. Erst wenn das Programm in den ersten Haltepunkt gelaufen ist, kann über das Symbol in Einzelschritten durch die nachfolgenden Zeilen debuggt werden.

Hinter einigen Befehlszeilen verbergen sich mehrere einzelne Befehle. Dadurch kann es passieren, dass zwei oder mehr Einzelschritte abgearbeitet werden, bevor im Eingabefenster die Schrittanzeige weiterspringt. Die aktuelle Position wird über einen kleinen Pfeil am linken PLC Editorfenster angezeigt.

Bei Betätigung des Symbols läuft das Programm bis zum nächsten Haltepunkt weiter.

Soll das Programm wieder frei laufen, so wird das Symbol betätigt.

## 2.5 PLC Konfiguration

Über das Symbol wird der PLC Konfigurationsdialog geöffnet. Hier können einige grundsätzliche Einstellungen für die PLC vorgenommen werden, auf die nachfolgend eingegangen wird.

**Überwachung der Zykluszeit**

Diese Funktion überwacht die max. Bearbeitungszeit für einen PLC Zyklus. Somit können ungewollt programmierte Dauerschleifen im PLC Programm abgefangen werden. Die Zeit ist in 5ms – Schritten bis zu max. 1s einstellbar. Im Falle einer Überschreitung wird im Frequenzumrichter der Fehler 22.4 ausgelöst.

**ParameterBox Funktionsbaustein zulassen**

Soll im PLC Programm eine Visualisierung über die ParameterBox erfolgen, dann muss diese Option aktiviert sein (Siehe Kapitel 3.5.6 ). Anderenfalls erzeugen die entsprechenden Funktionsblöcke beim Start des Frequenzumrichters einen Compiler Fehler.

**Ungültige Steuerdaten**

Die PLC kann die über die möglichen Bussysteme eingehenden Steuerwörter auswerten. Jedoch kommen die Steuerwörter nur durch, wenn das Bit „PZD gültig“ (Bit 10) gesetzt ist. Sollen auch nicht USS Protokoll konforme Steuerwörter von der PLC ausgewertet werden können, dann muss diese Option aktiviert sein. Bit 10 im ersten Wort wird dann nicht mehr abgefragt.

**Warmstart nach Fehler**

Alle Variablen werden beim Start der PLC immer mit „0“ oder ihrem Initialisierungswert geladen. Dabei ist es egal ob der Start nach einem Stopp, Programmdownload oder PLC Fehler erfolgt.

Über diese Option „Warmstart nach Fehler“ wird bei einem Warmstart der Inhalt der Variablen nicht verändert. Ein Warmstart erfolgt nach einem PLC Stopp Kommando oder einem PLC Fehler.

**Systemzeit beim Haltepunkt nicht anhalten**

Während des Debuggings, wenn die PLC im Haltepunkt oder sich im Einzelschrittmode befindet, wird die Systemzeit angehalten. Die Systemzeit bildet die Grundlage für alle Timer in der PLC.

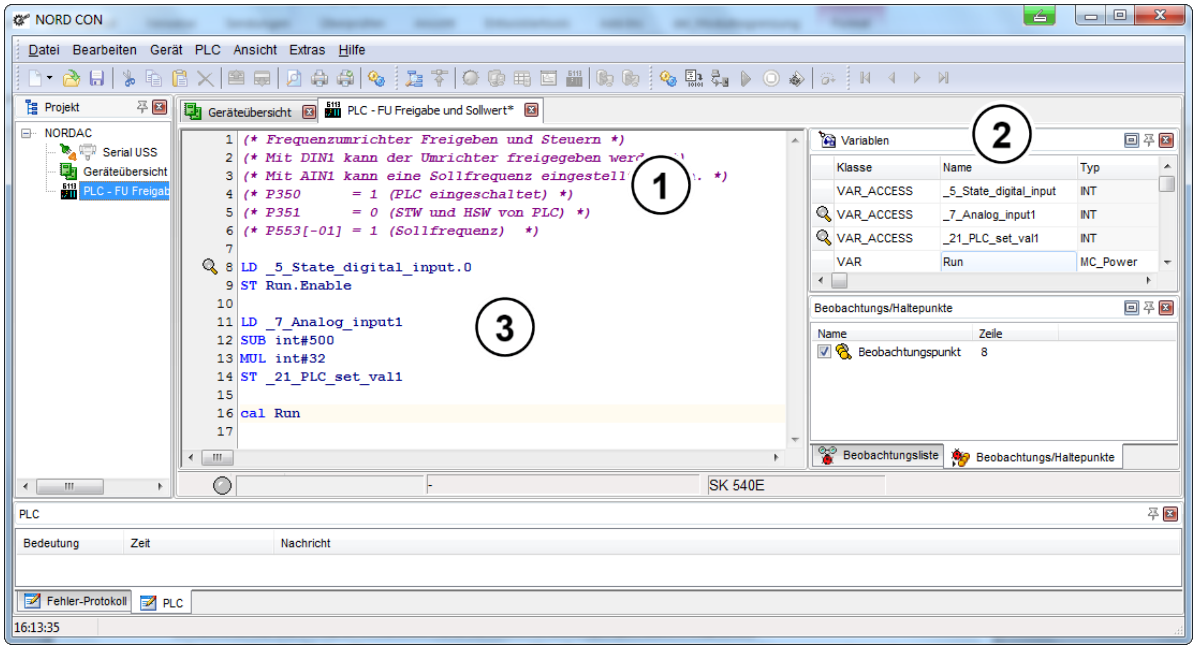
Soll die Systemzeit auch während des Debuggings weiterlaufen, dann ist diese Funktion zu aktivieren.

## 2.6 PLC Programmstart

Der Ablauf eines PLC Programms im Frequenzumrichter kann auf verschiedene Weise gestartet werden:

- **NORDCON**: Betätigen des Symbols in der Menüleiste.
- Parameter **P350** Einstellung 1 „An“ - Programmstart erfolgt unmittelbar nach Betriebsbereitschaft der Steuerkarte des Frequenzumrichters – z.B. nach Einschalten der Spannungsversorgung
- Bei **Programmunterbrechung** durch P420/P480 [-xx] Einstellung 80 „PLC Stop“: durch Rücksetzen des Eingangs

## 2.7 Programmbeispiel




1. Kommentar zur Programmfunktion und für Rahmenbedingungen zum Programm

z.B. Umrichter für PLC – Steuerung parametrieren (Die Parametrierung ist im Umrichter vorzunehmen.):

| | |
|---|---|
| Parameter P350 = 1 | Ansteuerung des FU über PLC |
| Parameter P351 = 0 | Steuerwort und Hauptsollwert von der PLC |
| Parameter P553[-01] = 1 | Der PLC Sollwert 1 wird als Sollfrequenz definiert |

2. Definition der verwendeten Variablen und Prozesswerte in der PLC

| | |
|---|---|
| _5_State_digital_input | Status der Digitaleingänge |
| _7_Analog_input1 | Wert Analogeingang1 |
| _21_PLC_set_val1 | PLC – Sollwert 1 |
| Run | Instanz des Motion Control Bausteins „MC_Power“ (Name frei wählbar) |

3. Programmablauf

| Befehl | Bedeutung | Beispiel |
|---|---|---|
| LD _5_State_digital_input.0 | Liest den Status der Digitaleingänge aus und lädt den Wert des Digitaleingang 1 in den Akku | Digitaleingang 1: high → Akku = 1 |
| ST Run.Enable | Speichert den Wert aus dem Akku in den durch die Variable Run aufgerufenen Motion Control-block als Wert für die Freigabe. | Freigabe = 1 |
| LD _7_Analog_input1 | Liest den Wert von Analogeingang 1 (Normierung: 10 V = 1000) und lädt ihn in den Akku | AIn1 = 2.5 V → Akku = 250 |
| SUB int#500 | Subtrahiert den Integerwert 500 vom Akku (Verschieben des 0-Punkt von 0 auf 5 V) | 250 - 500 = Akku = - 250 |
| MUL int#32 | Multipliziert den Integerwert 32 mit dem Akkuwert (→ Normierung des Sollwertes auf den Wertebereich des FU (100 % = $4000_{hex} \approx 16000_{dez}$). | -250 * 32 = Akku = - $8000_{dez}$ |
| ST _21_PLC_set_val1 | Speichert den Akku in die Prozessvariable PLC-Sollwert1. | $-8000_{dez}$ → Sollwert - 50 %<br>(Sollwert 50 % der Maximalfrequenz in Drehrichtung links) |
| Cal Run | Der Motion Control – Funktionbaustein für das Setzen der Umrichterendstufe wird ausgeführt | Endstufe ist freigegeben. |

# 3 AWL (Instruction List, IL)

## 3.1 Allgemein

### 3.1.1 Datentypen

Die PLC unterstütz die nachfolgend aufgeführten Datentypen.

| Name | Benötigter Speicherplatz | Wertebereich |
|---|---|---|
| BOOL | 1 Bit | 0 bis 1 |
| BYTE | 1 Byte | 0 bis 255 |
| INT | 2 Byte | -32768 bis 32767 |
| DINT | 4 Byte | -2.147.483.648 bis 2.147.483.647 |
| LABEL_ADDRESS | 2 Byte | Sprungmarken |

Tabelle 2 Datentypen

### 3.1.2 Literale

Zur besseren Übersicht ist es möglich Konstanten aller Datentypen in verschiedenen Darstellungsformen einzugeben. In nachfolgender Tabelle ist eine Übersicht über alle möglichen Varianten enthalten.

| Literal | Beispiel | Zahl in dezimaler Darstellung |
|---|---|---|
| **Bool** | FALSE | 0 |
| | TRUE | 1 |
| | BOOL#0 | 0 |
| | BOOL#1 | 1 |
| **Dual (Basis 2)** | 2#01011111 | 95 |
| | 2#0011_0011 | 51 |
| | BYTE#2#00001111 | 15 |
| | BYTE#2#0001_1111 | 31 |
| **Oktal (Basis 8)** | 8#0571 | 377 |
| | 8#05_71 | 377 |
| | BYTE#8#10 | 8 |
| | BYTE#8#111 | 73 |
| | BYTE#8#1_11 | 73 |
| **Hexadezimal (Basis 16)** | 16#FFFF | -1 |
| | 16#0001_FFFF | 131071 |
| | INT#16#1000 | 4096 |
| | DINT#16#0010_2030 | 1056816 |
| **Ganzzahlige (Basis 10)** | 10 | 10 |
| | -10 | -10 |
| | 10_000 | 10000 |
| | INT#12 | 12 |
| | DINT#-100000 | -100000 |
| **Zeit** | TIME#10s50ms | 10,050 Sekunden |
| | T#5s500ms | 5,5 Sekunden |
| | TIME#5.2s | 5,2 Sekunden |
| | TIME#5D10H15M | 5Tage+10Stunden+15Minuten |
| | T#1D2H30M20S | 1Tag+2Stunden+30Minuten+20Sekunden |

Tabelle 3 Numerische Literale

### 3.1.3 Kommentare

Für die spätere Lesbarkeit des PLC – Programmes ist es empfehlenswert Programmabschnitte mit Erklärungen zu versehen. Diese Kommentare werden im Anwenderprogramm beginnend durch die Zeichenfolge „(*“ und abschließend durch „*)“ gemäß nachfolgenden Beispielen gekennzeichnet. Die Kommentare werden nicht mit in den Frequenzumrichter übertragen.

```
(* Kommentar über einem Programmblock *)
LD 100 (* Kommentar hinter einem Befehl *)
ADD 20
```

### 3.1.4 Sprungmarke

Mit Hilfe der Operatoren JMP, JMPC oder JMPCN (Siehe Kapitel 3.3 ) können ganze Programmteile übersprungen werden. Als Zieladresse wird eine Sprungmarke angegeben. Sie kann mit Ausnahme von Umlauten und „ß“ alle Buchstaben, die Zahlen 0 bis 9 und Unterstriche enthalten, andere Zeichen sind nicht zulässig. Über einen Doppelpunkt wird die Sprungmarke abgeschlossen. Sie kann für sich alleine stehen. Es kann sich in derselben Zeile, hinter der Sprungmarke, auch noch ein weiterer Befehl befinden.

Mögliche Varianten könnten wie folgt aussehen:

```
Sprungmarke:
LD 20

Das_Ist_eine_Sprungmarke:
ADD 10

MainLoop: LD 1000
```

Eine weitere Variant ist die Übergabe einer Sprungmarke als Variable. Dies Variable muss in der Variablentabelle als Typ LABLE_ADDRESS definiert werden, dann können in diese Variable Sprungmarken geladen werden. Hierüber lassen sich sehr einfach Zustandsmaschinen erzeugen, siehe unten

```
JMP Address_Var

Address_1:
    LD Address_2
    ST Address_Var
    JMP Ende

Address_2:
    LD Address_3
    ST Address_Var
    JMP Ende

Address_3:
    LD Address_1
    ST Address_Var

Ende:
```

### 3.1.5 Funktionsaufrufe

Der Editor unterstützt z.Z. ausschließlich folgende listenartige Form von Funktionsaufrufen. Die Funktion CTD wird dabei über die Instanz I_CTD aufgerufen. Die Ergebnisse werden in Variablen gespeichert. Die Bedeutung der im Folgenden verwendeten Funktionen ist weiter hinten im Handbuch erläutert.

- LD 10000
- ST I_CTD.PV
- LD LoadNewVar
- ST I_CTD.LD
- LD TRUE
- ST I_CTD.CD
- CAL I_CTD
- LD I_CTD.Q
- ST ResultVar
- LD I_CTD.CV
- ST CurrentCountVar

### 3.1.6 Bitweiser Zugriff auf Variablen

Für den Zugriff auf ein Bit aus einer Variablen oder Prozessvariablen, ist eine vereinfachte Schreibweise möglich.

| Befehl | Bedeutung |
|---|---|
| LD Var1.0 | lädt das Bit 0 von Var1 ins AE |
| ST Var1.7 | speichert den AE auf das Bit 7 von Var1 |
| EQ Var1.4 | vergleicht das AE mit dem Bit4 von Var1 |

Tabelle 4 Bitweiser Zugriff auf Variablen

## 3.2 Operatoren

### 3.2.1 Lade- und Speicheroperatoren

| Operator | Erläuterung |
|---|---|
| LD | Lade |
| LDN | Lade negiert (BOOL) |
| ST | Speicher |
| STN | Speicher negiert (BOOL) |

Tabelle 5 Überblick Lade- und Speicheroperationen

#### 3.2.1.1 LD

Lädt eine Konstante oder eine Variable in den AE bzw. in den Akku

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10 | Lädt die 10 als BYTE | | | |
| LD -1000 | Lädt die -1000 als INT | | | |
| LD Value1 | Lädt die Variable Value1 | | | |

Tabelle 6 LD

#### 3.2.1.2 LDN

Lädt eine boolesche Variable negiert in den AE

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | | | |
| **Befehle** | **Erläuterung** | | | |
| LDN Value1 | Value1 = TRUE → AE = FALSE | | | |
| ST Value2 | Speichere auf Value2 = FALSE | | | |

Tabelle 7 LDN

#### 3.2.1.3 ST

Speichert den Inhalt des AE/Akkus auf eine Variable ab. Die abzuspeichernde Variable muss zu dem vorher geladenen und verarbeiteten Datentyp passen.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 100 | Lädt den Wert 100 | | | |
| ST Value1 | Akku Inhalt 100 wird in Value1 abgespeichert | | | |

Tabelle 8 ST

#### 3.2.1.4 STN

Speichert den Inhalt des AE auf eine Variable ab und negiert ihn. Die abzuspeichernde Variable muss zu dem vorher geladenen und verarbeiteten Datentyp passen.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | | | |
| **Befehle** | **Erläuterung** | | | |
| LD Value1 | Value1 = TRUE → AE = TRUE | | | |
| STN Value2 | Speichere auf Value2 = FALSE | | | |

Tabelle 9 STN

## 3.2.2 Arithmetische Operatoren

| Operator | Erläuterung |
|---|---|
| ABS | Absoluter Betrag |
| ADD | Addition |
| DIV | Division |
| LIMIT | Begrenzer |
| MAX | Ermittelt von zwei Zahlen die größere Zahl |
| MIN | Ermittelt von zwei Zahlen die kleinere Zahl |
| MUX | Multiplexer |
| MOD | Modulo Operation |
| MUL | Multiplikation |
| SUB | Subtraktion |

Tabelle 10 Übersicht arithmetische Operatoren

**HINWEIS**

Einzelne der folgenden Operatoren können auch weiterführende Befehle beinhalten. Diese sind in Klammern hinter den Operator zu setzen.

Dabei ist zu beachten, dass hinter der eröffnenden Klammer ein Leerzeichen stehen muss. Die schließende Klammer ist auf eine separate Programmzeile zu setzen.

```
LD Var1
ADD( Var2
SUB Var3
)
```

### 3.2.2.1 ABS

Bildet aus dem Akku den absoluten Betrag.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD -10 | Lädt den Wert -10 | | | |
| ABS | Akku = 10 | | | |
| ST Value1 | Speichert den Wert 10 in Value1 ab | | | |

Tabelle 11 ABS

### 3.2.2.2 ADD und ADD(

Addiert vorzeichenrichtig Variablen und Konstanten miteinander. Der erste Wert zur Addition befindet sich im Akku und der zweite wird mit dem ADD Befehl geladen oder er befindet sich innerhalb der Klammer. Es können auch mehrere Variablen oder Konstanten an den ADD Befehl angefügt werden.

Bei der Klammer Addition wird der Akku mit dem Ergebnis des Klammerausdrucks addiert. Es sind bis zu 6 Klammerebenen möglich.

Die zu addierenden Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>ADD 204 | Addition zweier Konstanten | | | |
| LD 170<br>ADD Var1, Var2 | Addition einer Konstanten und 2 Variablen.<br>$170_{dez}$ + Var1 + Var2 | | | |
| LD Var1<br>ADD( Var2<br>SUB Var3<br>) | Var1 + ( Var2 - Var3 ) | | | |

Tabelle 12 ADD und ADD(

### 3.2.2.3 DIV und DIV(

Dividiert den Akku durch den Operanden. Bei Divisionen durch null wird das maximal mögliche Ergebnis in den Akku eingetragen, z.B. bei einer Division mit INT Werten ist das der Wert 0x7FFF oder wenn der Divisor negativ ist dann ist es der Wert 0x8000.

Bei der Klammer Division wird der Akku durch das Ergebnis des Klammerausdrucks dividiert. Es sind bis zu 6 Klammerebenen möglich.

Die zu dividierenden Werte müssen demselben Variablentyp angehören.

Nachkommastellen des Ergebnisses werden abgeschnitten. Werden die Nachkommastellen benötigt, kann über die Modulo Operation (MOD) der Rest der Division ermittelt und mit diesem weitergerechnet werden.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>DIV 2 | Division zweier Konstanten | | | |
| LD 170<br>DIV Var1, Var2 | Division einer Konstanten und 2 Variablen.<br>($170_{dez}$ : Var1) : Var2 | | | |
| LD Var1<br>DIV( Var2<br>SUB Var3<br>) | Dividiere Var1 durch den Inhalt der Klammer<br>Var1 : ( Var2 - Var3 ) | | | |

Tabelle 13 DIV und DIV(

### 3.2.2.4 LIMIT

Der Befehl begrenzt den im Akku stehenden Wert auf die übergebenen min. und max. Werte. Bei Überschreitung wird im Akku der max. Wert eingetragen und bei Unterschreitung der min. Wert. Liegt der Wert zwischen den Limits, so erfolgt keine Beeinflussung.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10 | Lädt den Wert 10 in den Akku | | | |
| LIMIT 20,30 | Der Wert wird mit den Grenzen 20 und 30 verglichen. Der Wert im Akku ist kleiner, der Akku wird mit 20 überschrieben | | | |
| ST Value1 | Speichert den Wert 20 in Value1 ab | | | |

Tabelle 14 LIMIT

### 3.2.2.5 MAX

Dieser Befehl ermittelt den maximalen Wert von zwei Variablen oder Konstanten. Dazu wird der aktuelle Akku Inhalt mit dem im MAX Befehl übergebenen Wert verglichen. Der größere von beiden Werten befindet sich nach dem Befehl im Akku.

Beide Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 100 | Lade 100 in den Akku | | | |
| MAX 200 | Vergleiche mit dem Wert 200 | | | |
| ST Value2 | Speichere den Wert 200 in Value2 (weil größter Wert) | | | |

Tabelle 15 MAX

### 3.2.2.6 MIN

Dieser Befehl ermittelt den minimalen Wert von zwei Variablen oder Konstanten. Dazu wird der aktuelle Akku Inhalt dem im MIN Befehl übergebenen Wert verglichen. Der kleinere von beiden Werten befindet sich nach dem Befehl im Akku.

Beide Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 100 | Lade 100 in den Akku | | | |
| MIN 200 | Vergleiche mit dem Wert 200 | | | |
| ST Value2 | Speichere den Wert 100 in Value2 (weil kleinerer Wert) | | | |

Tabelle 16 MIN

### 3.2.2.7 MUX

Über einen Index, der sich vor dem Befehl im Akku befindet, können verschiedene Konstanten oder Variablen selektiert werden. Der erste Wert wird über den Index 0 angesprochen. Der ausgewählte Wert wird in den Akku geladen. Die Anzahl der Werte ist nur durch den Programmspeicher limitiert.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 1 | Wähle das gewünschte Element aus | | | |
| MUX 10,20,30,40,Value1 | MUX Befehl mit 4 Konstanten und einer Variable | | | |
| ST Value2 | Speichere den Wert 20 in Value2 | | | |

Tabelle 17 MUX

### 3.2.2.8 MOD und MOD(

Der Akku wird durch eine oder mehrere Variablen oder Kostanten dividiert, der Rest der Division steht als Ergebnis im Akku.

Bei der Klammer Modulo wird der Akku durch das Ergebnis des Klammerausdrucks dividiert und daraus der Modulo gebildet. Es sind bis zu 6 Klammerebenen möglich.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 25 | Lade den Dividenden | | | |
| MOD 20 | Division 25/20 → Modulo = 5 | | | |
| ST Var1 | Speicher das Ergebnis 5 in Var1 | | | |
| | | | | |
| LD 25 | Lade den Dividenden | | | |
| MOD( Var1 | Ergebnis = 25/(Var1 + 10) → Modulo in den Akku | | | |
| ADD 10 | | | | |
| ) | | | | |
| ST Var3 | Speichere das Ergebnis 10 in Var3 | | | |

Tabelle 18 MOD und MOD(

### 3.2.2.9 MUL und MUL(

Multiplikation des Akkus mit einer oder mehreren Variablen oder Konstanten.

Bei der Klammer Multiplikation wird der Akku mit dem Ergebnis des Klammerausdrucks multipliziert. Es sind bis zu 6 Klammerebenen möglich.

Beide Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 25 | Lade den Multiplikator | | | |
| MUL Var1, Var2 | 25 * Var1 * Var2 | | | |
| ST Var2 | Speichere Ergebnis | | | |
| | | | | |
| LD 25 | Lade den Multiplikator | | | |
| MUL( Var1 | Ergebnis = 25*(Var1 + Var2) | | | |
| ADD Var2 | | | | |
| ) | | | | |
| ST Var3 | Speichere das Ergebnis als Variable Var3 | | | |

Tabelle 19 MUL und MUL(

### 3.2.2.10 SUB und SUB(

Subtrahiert den Akku mit einer oder mehreren Variablen oder Konstanten.

Bei der Klammer Subtraktion wird der Akku mit dem Ergebnis des Klammerausdrucks subtrahiert. Es sind bis zu 6 Klammerebenen möglich.

Die zu subtrahierenden Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10 | | | | |
| SUB Value1 | Ergebnis = 10 - Value1 | | | |
| ST Value1 | | | | |
| LD 20 | | | | |
| SUB Value1,Value2,30 | Ergebnis = 20 - Value1 - Value2 - 30 | | | |
| ST Value1 | | | | |
| LD 20 | | | | |
| SUB( 6 | Subtrahiere 20 mit den Inhalt der Klammer | | | |
| AND 2 | | | | |
| ) | Ergebnis = 20 - (6 AND 2) | | | |
| ST Value1 | Speichere das Ergebnis 18 in Value1 | | | |

Tabelle 20 SUB und SUB(

### 3.2.3 Erweiterte mathematische Operatoren

| Operator | Erläuterung |
|---|---|
| EXP | Exponentialfunktion |
| LOG | Logarithmus Basis 10 |
| LN | Logarithmus Basis e |
| SQRT | Wurzel |
| | |
| COS, ACOS | Trigonometrische Operatoren |
| SIN, ASIN | |
| TAN, ATAN | |

Tabelle 21 Überblick Erweiterte mathematische Operatoren

**HINWEIS**

Die hier aufgeführten Operatoren sind sehr rechnenintensiv. Es kann zu deulich längeren Laufzeiten des PLC Programmes kommen.

#### 3.2.3.1 EXP

Bildet aus dem Akku die Exponentialfunktion zur Basis der Eulerschen Zahl (2,718). Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,002 muss als 1002 eingegeben werden.

$$Akku = e^{\left(\frac{Akku}{1000}\right)} \times 1000$$

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | | X |
| **Befehle** | **Erläuterung** | | | |
| LD 1000<br>EXP<br>ST Var1 | Ergebnis = 2718 | | | |

Tabelle 22 EXP

#### 3.2.3.2 LOG

Bildet aus dem Akku den Logarithmus zur Basis 10. Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,000 muss als 1000 eingegeben werden.

$$Akku = \log_{10}\left(\frac{Akku}{1000}\right) \times 1000$$

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | | X |
| **Befehle** | **Erläuterung** | | | |
| LD 1234<br>LOG<br>ST Var1 | Ergebnis = 91 | | | |

Tabelle 23 LOG

### 3.2.3.3 LN

Logarithmus zur Basis e (2,718). Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,000 muss als 1000 eingegeben werden.

$$Akku = \ln\left(\frac{Akku}{1000}\right) \times 1000$$

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | | X |
| **Befehle** | **Erläuterung** | | | |
| LD 1234<br>LN<br>ST Var1 | <br>Ergebnis = 210<br>Speichere den Wert 210 in Var1 | | | |

Tabelle 24 LN

### 3.2.3.4 SQRT

Bildet aus dem Akku die Quadratwurzel. Es können 3 Nachkommastellen angegeben werden, d.h. eine 1,000 muss als 1000 eingegeben werden.

$$Akku = \sqrt{\left(\frac{Akku}{1000}\right)} \times 1000$$

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | | X |
| **Befehle** | **Erläuterung** | | | |
| LD 1234<br>SQRT<br>ST Var1 | <br>Ergebnis = 1110<br>Speichere den Wert 1110 in Var1 | | | |

Tabelle 25 SQR

### 3.2.3.5 COS, ACOS, SIN, ASIN, TAN, ATAN

Berechnung der jeweiligen mathematischen Funktion. Der zu berechnende Wert muss im Akku als Bogenmaß vorliegen. Die Skalierung entspricht 1 = 1000.

Umrechnung: Winkel in Bogenmaß = (Winkel in Grad * PI / 180)*1000
z.B. ein Winkel von 90° wird wie folgt umgerechnet → 90° * 3.14 / 180) *1000 = 1571

$$Akku = \sin\left(\frac{Akku}{1000}\right) \times 1000 \qquad Akku = \cos\left(\frac{Akku}{1000}\right) \times 1000 \qquad Akku = \tan\left(\frac{Akku}{1000}\right) \times 1000$$

$$Akku = a\sin\left(\frac{Akku}{1000}\right) \times 1000 \qquad Akku = a\cos\left(\frac{Akku}{1000}\right) \times 1000 \qquad Akku = a\tan\left(\frac{Akku}{1000}\right) \times 1000$$

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | | X |
| **Befehle** | **Erläuterung** | | | |
| LD 1234<br>SIN<br>ST Var1 | <br><br>Speichere den Wert 943 in Var1 | | | |

Tabelle 26 Trigonometrische Funktionen

## 3.2.4 Bit Operationen

| Operator | Erläuterung |
|---|---|
| AND | UND |
| ANDN | UND mit negiertem Operanden |
| NOT | Negation |
| OR | ODER |
| ORN | ODER mit negiertem Operanden |
| ROL | Links Rotieren |
| ROR | Rechts Rotieren |
| SHL | Links Schieben |
| SHR | Rechts Schieben |
| S und R | Setzen & Rücksetzen |
| XOR | Exklusiv ODER |
| XORN | Exklusiv ODER mit negiertem Operanden |

Tabelle 27 Übersicht Bit Operationen

### 3.2.4.1 AND und AND(

Bitweise UND Verknüpfung des AE/Akkus mit einer oder zwei Variablen oder Konstanten.

Bitweise UND(…) Verknüpfung mit dem AE/Akku und dem AE/Akku welches zuvor in der Klammer gebildet wurde. Es sind bis zu 6 Klammerebenen möglich.

Alle Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |

| Befehle | Erläuterung |
|---|---|
| LD 170<br>AND 204 | AND Verknüpfung zwischen 2 Konstanten Akku = 136 (Siehe Beispiel unter der Tabelle) |
| LD 170<br>AND Var1, Var2 | Verknüpfung zwischen einer Konstanten und 2 Variablen.<br>Akku = $170_{dez}$ AND Var1 AND Var2 |
| LD Var1<br>AND ( Var2<br>OR Var3<br>) | AE/Akku = Var1 AND ( Var2 OR Var3 ) |

| AE/Akku | Operand | Ergebnis |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tabelle 28 AND und AND(

Beispiel: **$170_{dez}$** ($1010\ 1010_{bin}$) **AND $204_{dez}$** ($1100\ 1100_{bin}$) **=** ($1000\ 1000_{bin}$) **$136_{dez}$**

### 3.2.4.2 ANDN und ANDN(

Bitweise UND Verknüpfung des AE/Akkus mit einem negierten Operanden.

Bitweise UND (…) Verknüpfung mit dem AE/Akku und dem negierten Ergebnis der Klammer. Es sind bis zu 6 Klammerebenen möglich.

Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |

| **Befehle** | **Erläuterung** |
|---|---|
| LD 2#0000_1111<br>ANDN 2#0011_1010 | ANDN Verknüpfung zwischen 2 Konstanten Akku = 2#0000_0101 |
| LD 170<br>ANDN Var1, Var2 | Verknüpfung zwischen einer Konstanten und 2 Variablen.<br>Akku = 170d ANDN Var1 ANDN Var2 |
| LD Var1<br>ANDN ( Var2<br>OR Var3<br>) | AE/Akku = Var1 ANDN ( Var2 OR Var3 ) |

| AE/Akku | Operand | Ergebnis |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Tabelle 29 ANDN und ANDN(

### 3.2.4.3 NOT

Bitweise Negation des Akkus.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |

| **Befehle** | **Erläuterung** |
|---|---|
| LD BYTE#10 | Lade In den AKKU den Wert $10_{dez}$ im Format Byte |
| NOT | Der Wert wird auf Bit – Ebene aufgelöst (0000 1010), bitweise negiert (1111 0101) und wieder in einen Dezimalwert gewandelt, Ergebnis = $245_{dez}$ |
| ST Var3 | Speichere das Ergebnis als Variable Var3 |

Tabelle 30 NOT

### 3.2.4.4 OR und OR(

Bitweise ODER Verknüpfung des AE/Akkus mit einer oder zwei Variablen oder Konstanten.

Bitweise ODER(…) Verknüpfung mit dem AE/Akku und dem AE/Akku welches zuvor in der Klammer gebildet wurde. Es sind bis zu 6 Klammerebenen möglich.

Alle Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |

| Befehle | Erläuterung |
|---|---|
| LD 170<br>OR 204 | OR Verknüpfung zwischen 2 Konstanten |
| LD 170<br>OR Var1, Var2 | Verknüpfung zwischen einer Konstanten und 2 Variablen.<br>Akku = 170d OR Var1OR Var2 |
| LD Var1<br>OR ( Var2<br>AND Var3<br>) | AE/Akku = Var1 OR ( Var2 AND Var3 ) |

| AE/Akku | Operand | Ergebnis |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Tabelle 31 OR und OR(

### 3.2.4.5 ORN und ORN(

Bitweise ODER Verknüpfung des AE/Akkus mit einem negierten Operanden.

Bitweise ODER (…) Verknüpfung mit dem AE/Akku und dem negierten Ergebnis der Klammer. Es sind bis zu 6 Klammerebenen möglich.

Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |

| Befehle | Erläuterung |
|---|---|
| LD 2#0000_1111<br>ORN 2#0011_1010 | ORN Verknüpfung zwischen 2 Konstanten<br>Akku = 2#1100_1111 |
| LD 170<br>ORN Var1, Var2 | Verknüpfung zwischen einer Konstanten und 2 Variablen.<br>Akku = 170d ORN Var1 ORN Var2 |
| LD Var1<br>ORN ( Var2<br>OR Var3<br>) | AE/Akku = Var1 ORN ( Var2 OR Var3 ) |

| AE/Akku | Operand | Ergebnis |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Tabelle 32 ANDN und ANDN(

### 3.2.4.6 ROL

Bitweise Linksrotation des Akkus. Dabei wird der Inhalt des Akkus um n mal nach links verschoben, wobei das links Bit wieder rechts reingeschoben wird.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 175 | Lädt den Wert 1010_1111 | | | |
| ROL 2 | Akku Inhalt wird 2 mal nach links rotiert | | | |
| ST Value1 | Speichert den Wert 1011_1110 in Value1 ab | | | |

Tabelle 33 ROL

### 3.2.4.7 ROR

Bitweise Rechtsrotation des Akkus. Dabei wird der Inhalt des Akkus um n mal nach rechts verschoben, wobei das rechte Bit wieder links reingeschoben wird.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 175 | Lädt den Wert 1010_1111 | | | |
| ROR 2 | Akku Inhalt wird 2 mal nach rechts rotiert | | | |
| ST Value1 | Speichert den Wert 1110_1011 in Value1 ab | | | |

Tabelle 34 ROR

### 3.2.4.8 SHL

Bitweises Linksschieben des Akkus. Dabei wird der Inhalt des Akkus um n mal nach links verschoben, die rausgeschobenen Bits sind verloren.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 175 | Lädt den Wert 1010_1111 | | | |
| SHL 2 | Akku Inhalt wird 2 mal nach links verschoben | | | |
| ST Value1 | Speichert den Wert 1011_1100 in Value1 ab | | | |

Tabelle 35 SHL

### 3.2.4.9 SHR

Bitweises Rechtsschieben des Akkus. Dabei wird der Inhalt des Akkus um n mal nach rechts verschoben, die rausgeschobenen Bits sind verloren.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 175 | Lädt den Wert 1010_1111 | | | |
| SHR 2 | Akku Inhalt wird 2 mal nach rechts verschoben | | | |
| ST Value1 | Speichert den Wert 0010_1011 in Value1 ab | | | |

Tabelle 36 SHR

### 3.2.4.10 S und R

Setzen und Rücksetzen einer booleschen Variablen, wenn das vorherige Verknüpfungsergebnis (das AE) TRUE war.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | | | |
| **Befehle** | **Erläuterung** | | | |
| LD TRUE | Lädt das AE mit TRUE | | | |
| S Var1 | VAR1 wird TRUE gesetzt | | | |
| R Var1 | VAR1 wird FALSE gesetzt | | | |

Tabelle 37 S und R

### 3.2.4.11 XOR und XOR(

Bitweise „Exklusiv Oder“ Verknüpfung zwischen dem AE/Akku und ein bis zwei Variablen oder Konstanten. Der erste Wert befindet sich im AE/Akku, der zweite wird mit dem Befehl geladen oder er befindet sich innerhalb der Klammer. Es sind bis zu 6 Klammerebenen möglich.

Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 2#0000_1111<br>XOR 2#0011_1010 | XOR Verknüpfung zwischen 2 Konstanten<br>Akku = 2#0011_0101 | | | |
| LD 170<br>XOR Var1, Var2 | Verknüpfung zwischen einer Konstanten und 2 Variablen.<br>Akku = 170d XOR Var1 XOR Var2 | | | |
| LD Var1<br>XOR ( Var2<br>OR Var3<br>) | AE/Akku = Var1 XOR ( Var2 OR Var3 ) | | | |

| AE/Akku | Operand | Ergebnis |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Tabelle 38 XOR oder XOR(

### 3.2.4.12 XORN und XORN(

Bitweise Exclusiv ODER Verknüpfung des AE/Akkus mit einem negierten Operanden.

Bitweise Exclusiv ODER (…) Verknüpfung mit dem AE/Akku und dem negierten Ergebnis der Klammer. Es sind bis zu 6 Klammerebenen möglich.

Die zu verknüpfenden Werte müssen demselben Variablentyp angehören.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |

| Befehle | Erläuterung |
|---|---|
| LD 2#0000_1111<br>XORN 2#0011_1010 | XORN Verknüpfung zwischen 2 Konstanten<br>Akku = 2#1100_1010 |
| LD 170<br>XORN Var1, Var2 | Verknüpfung zwischen einer Konstanten und 2 Variablen.<br>Akku = 170d XORN Var1 XORN Var2 |
| LD Var1<br>XORN ( Var2<br>OR Var3<br>) | AE/Akku = Var1 XORN ( Var2 OR Var3 ) |

| AE/Akku | Operand | Ergebnis |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tabelle 39 XORN und XORN(

## 3.2.5 Vergleichs Operatoren

| Operator | Erläuterung |
|---|---|
| EQ | Gleich |
| GE | Größer oder Gleich |
| GT | Größer |
| LE | Kleiner oder Gleich |
| LT | Kleiner |
| NE | Ungleich |

Tabelle 40 Überblick Vergleichsoperatoren

### 3.2.5.1 EQ

Vergleicht den Inhalt vom Akku mit einer Variablen oder Konstanten. Sind die Werte gleich, dann wird das AE auf TRUE gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD Value1 | Value1 = 5 | | | |
| EQ 10 | AE = Ist 5 gleich 10 ? | | | |
| JMPC NextStep | AE = FALSE → Programm springt nicht | | | |
| ADD 1 | | | | |
| NextStep: | | | | |
| ST Value1 | | | | |

Tabelle 41 EQ

### 3.2.5.2 GE

Vergleicht den Inhalt vom Akku mit einer Variablen oder Konstanten. Ist der Wert im Akku größer oder gleich der Variablen oder Konstanten, dann wird das AE auf TRUE gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD Value1 | Value1 = 5 | | | |
| GE 10 | Ist 5 größer oder gleich 10? | | | |
| JMPC NextStep | AE = FALSE → Programm springt nicht | | | |
| ADD 1 | | | | |
| NextStep: | | | | |
| ST Value1 | | | | |

Tabelle 42 GE

### 3.2.5.3 GT

Vergleicht den Inhalt vom Akku mit einer Variablen oder Konstanten. Ist der Wert im Akku größer als die Variable oder Konstante, dann wird das AE auf TRUE gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD Value1 | Value1 = 12 | | | |
| GT 8 | Ist 12 größer 8? | | | |
| JMPC NextStep | AE = TRUE → Programm springt | | | |
| ADD 1 | | | | |
| NextStep: | | | | |
| ST Value1 | | | | |

Tabelle 43 GT

### 3.2.5.4 LE

Vergleicht den Inhalt vom Akku mit einer Variable oder Konstanten. Ist der Wert im Akku kleiner oder gleich der Variablen oder Konstanten, dann wird das AE auf TRUE gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD Value1 | Value1 = 5 | | | |
| LE 10 | Ist 5 kleiner oder gleich 10? | | | |
| JMPC NextStep | AE = TRUE → Programm springt | | | |
| ADD 1 | | | | |
| NextStep: | | | | |
| ST Value1 | | | | |

Tabelle 44 LE

### 3.2.5.5 LT

Vergleicht den Inhalt vom Akku mit einer Variablen oder Konstanten. Ist der Wert im Akku kleiner als die Variable oder Konstante, dann wird das AE auf TRUE gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD Value1 | Value1 = 12 | | | |
| LT 8 | Ist 12 kleiner 8 ? | | | |
| JMPC NextStep | AE = FALSE → Programm springt nicht | | | |
| ADD 1 | | | | |
| NextStep: | | | | |
| ST Value1 | | | | |

Tabelle 45 LT

### 3.2.5.6 NE

Vergleicht den Inhalt vom Akku mit einer Variablen oder Konstanten. Ist der Wert im Akku ungleich der Variablen oder Konstanten, dann wird das AE auf TRUE gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD Value1<br>NE 10<br>JMPC NextStep<br>ADD 1<br>NextStep:<br>ST Value1 | Value1 = 5<br>Ist 5 ungleich 10 ?<br>AE = TRUE → Programm springt | | | |

Tabelle 46 NE

## 3.3 Sprünge

| Operator | Erläuterung |
|---|---|
| JMP | Springe |
| JMPC | Springe wenn AE=TRUE |
| JMPCN | Springe wenn AE=FALSE |

Tabelle 47 Überblick Sprünge

### 3.3.1 JMP

Unbedingter Sprung zu einer Sprungmarke.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | - - - | - - - | - - - | - - - |
| **Befehle** | **Erläuterung** | | | |
| JMP NextStep<br>ADD 1<br>NextStep:<br>ST Value1 | Unbedingter Sprung zu NextStep | | | |

Tabelle 48 JMP

### 3.3.2 JMPC

Bedingter Sprung (Jump Conditional) zu einer Sprungmarke. Ist das AE = TRUE dann springt die Anweisung JMPC zur angegebenen Sprungmarke.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>JMPC NextStep<br>ADD 1<br>NextStep:<br>ST Value1 | AE = TRUE → Programm springt | | | |

Tabelle 49 JMPC

### 3.3.3 JMPCN

Bedingter Sprung (Jump Conditional) zu einer Sprungmarke. JMPCN springt wenn das AE Register = FALSE ist. Ansonsten wird das Programm mit der nachfolgenden Anweisung fortgesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | X | X | X |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>JMPCN NextStep<br>ADD 1<br>NextStep:<br>ST Value1 | AE = TRUE → Programm springt nicht | | | |

Tabelle 50 JMPCN

## 3.4 Typkonvertierung

| Operator | Erläuterung |
|---|---|
| BYTE_TO_BOOL | Konvertierung BYTE zu BOOL |
| BOOL_TO_BYTE | Konvertierung BOOL zu BYTE |
| INT_TO_BYTE | Konvertierung INT zu BYTE |
| BYTE_TO_INT | Konvertierung BYTE zu INT |
| DINT_TO_INT | Konvertierung DINT zu INT |
| INT_TO_DINT | Konvertierung INT zu DINT |

Tabelle 51 Überblick Typenkonvertierung

### 3.4.1 BYTE_TO_BOOL

Konvertiert den Datentyp von BYTE zu BOOL. Solange das BYTE ungleich Null ist, gibt es immer ein TRUE als Konvertierungsergebnis.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | | |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>BYTE_TO_BOOL | AE = TRUE | | | |

Tabelle 52 BYTE_TO_BOOL

### 3.4.2 BOOL_TO_BYTE

Konvertiert den Datentyp AE von BOOL zu BYTE. Ist das AE gleich FALSE, dann wird der Akku auf 0 konvertiert. Ist das AE gleich TRUE, dann wird der Akku auf 1 konvertiert.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | X | | | |
| **Befehle** | **Erläuterung** | | | |
| LD TRUE<br>BOOL_TO_BYTE | Akku = 1 | | | |

Tabelle 53 BOOL_TO_BYTE

### 3.4.3 INT_TO_BYTE

Konvertiert den Datentyp von INT zu BYTE. Dabei wird der High Teil vom INT Wert nicht mit übernommen. Vorzeichen gehen verloren, da der Typ BYTE vorzeichenlos ist.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | X | |
| **Befehle** | **Erläuterung** | | | |
| LD 16#5008<br>INT_TO_BYTE | Akku = 8 | | | |

Tabelle 54 INT_TO_BYTE

### 3.4.4 BYTE_TO_INT

Konvertiert den Datentyp von BYTE zu INT. Das BYTE wird in den Low Teil des INT hineinkopiert und der High Teil vom INT wird 0 gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | X | | |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>BYTE_TO_INT | Akku = 10 | | | |

Tabelle 55 BYTE_TO_INT

### 3.4.5 DINT_TO_INT

Konvertiert den Datentyp von DINT zu INT. Bei DINT Werten die in den INT Bereich hineinpassen, wird vorzeichenrichtig konvertiert. Bei allen anderen Werten wird der High Teil vom DINT Wert nicht mit übernommen.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | | X |
| **Befehle** | **Erläuterung** | | | |
| LD DINT# 5000<br>DINT_TO_INT<br>LD DINT# -5000<br>DINT_TO_INT<br>LD DINT# 200.000<br>DINT_TO_INT | Akku = 5000<br>Akku = -5000<br>Akku = 3392 | | | |

Tabelle 56 DINT_TO_INT

### 3.4.6 INT_TO_DINT

Konvertiert den Datentyp von INT zu DINT. Das INT wird in den Low Teil des DINT hineinkopiert und der High Teil vom DINT wird 0 gesetzt.

| | BOOL | BYTE | INT | DINT |
|---|---|---|---|---|
| **Mögliche Datentypen** | | | X | |
| **Befehle** | **Erläuterung** | | | |
| LD 10<br>INT_TO_DINT | Akku = 10 | | | |

Tabelle 57 INT_TO_DINT

## 3.5 Funktionsblöcke

Funktionsblöcke sind kleinere Programme, die ihre Zustandswerte in internen Variablen ablegen können. Aus diesem Grund muss für jeden Funktionsblock eine eigene Instanz in der Variablenliste von NORDCON erzeugt werden. Soll z.B. ein Timer parallel 3 Zeiten überwachen, so muss er in der Variablenliste auch dreimal angelegt werden.

Aufruf eines Funktionsblocks: „**Cal** *Funktionsblock*“ (Bsp. **Cal** Timer)

**HINWEIS**

Funktionsblöcke müssen zwingen zyklisch aufgerufen werden! Dies ist wichtig, da die meisten Funktionen in den FB nicht in einem Zyklus abgearbeitet werden.

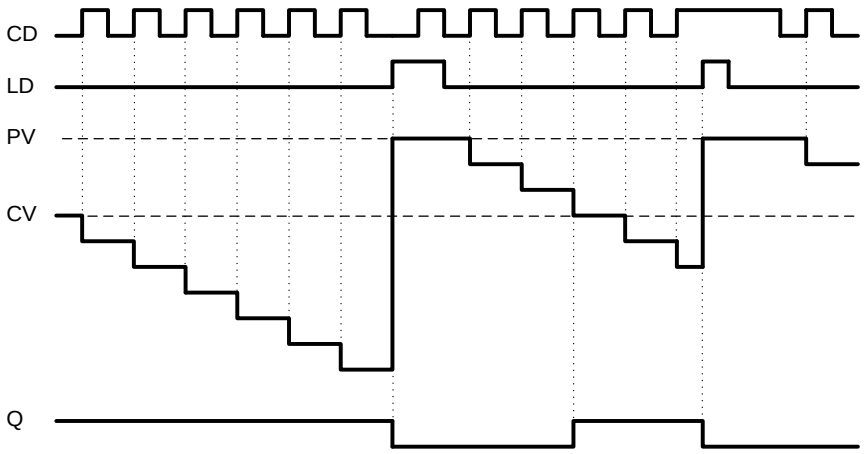
### 3.5.1 Standard

| Funktionsblock | Erläuterung |
|---|---|
| CTD | Abwärtszähler |
| CTU | Aufwärtszähler |
| CTUD | Auf- und Abwärtszähler |
| SR | Bistabile Funktion, Setzen dominant |
| RS | Bistabile Funktion, Rücksetzen dominant |
| R_TRIG | Flankenerkennung, steigende Flanke |
| F_TRIG | Flankenerkennung, fallende Flanke |
| TON | Einschaltverzögerung |
| TOF | Ausschaltverzögerung |
| TP | Zeitimpuls |

Tabelle 58 Überblick Standartbibliothek

### 3.5.1.1 CTD Abwärtszähler

Bei steigender Flanke an **CD** wird der Zähler des Funktionsblockes **CV** um eins verringert, solange CV größer als -32768 ist. Wenn **CV** kleiner oder gleich 0 ist, bleibt der Ausgang **Q** auf TRUE. Über **LD** kann der Zähler **CV** auf den in **PV** gespeicherten Wert gesetzt werden.

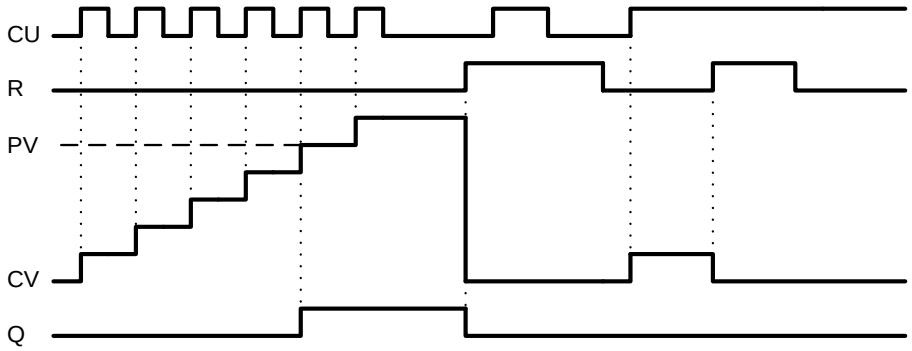



| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **CD** | Zählereingang | BOOL | **Q** | TRUE, wenn CV <= 0 | BOOL |
| **LD** | Lade Startwert | BOOL | **CV** | Aktueller Zählerstand | INT |
| **PV** | Startwert | INT | | | |

Tabelle 59 CTD Abwärtszähler

### 3.5.1.2 CTU Aufwärtszähler

Bei steigender Flanke an **CU,** wird der Zähler des Funktionsblockes **CV** um eins erhöht. **CV** kann bis auf den Wert 32767 gezählt werden. Solange **CV** größer oder gleich **PV** ist, bleibt der Ausgang **Q** auf TRUE. Über **R** kann der Zähler **CV** auf den Wert null zurückgesetzt werden.



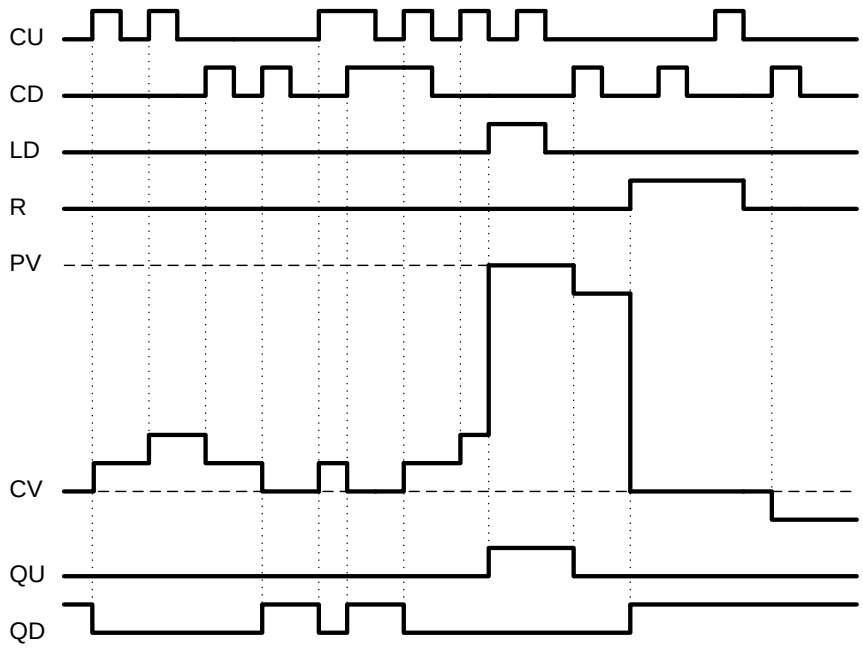

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **CU** | Zählereingang | BOOL | **Q** | TRUE, wenn CV >= PV | BOOL |
| **R** | Reset Zählerstand | BOOL | **CV** | Aktueller Zählerstand | INT |
| **PV** | Max. Zählerwert | INT | | | |

Tabelle 60 CTU Aufwärtszähler

### 3.5.1.3 CTUD Auf- und Abwärtszähler

Bei steigender Flanke an **CU** wird der Zähler CV um eins erhöht, solange CV kleiner als 32767 ist. Bei steigender Flanke an **CD** wird der Zähler **CV** um eins verringert, solange **CV** größer als -32768 ist. Über **R** kann der Zähler **CV** auf den Wert Null gesetzt werden. Über **LD** wird der in **PV** gespeicherte Wert in **CV** kopiert.

**LD** und **R** haben Vorrang gegenüber CU und CV. **PV** kann jederzeit verändert werden, **QU** bezieht sich immer auf den aktuell eingestellten Wert.




| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **CU** | Aufwärtszählen | BOOL | **QU** | TRUE, wenn CV >= PV | BOOL |
| **CD** | Abwärtszählen | BOOL | **QD** | TRUE, wenn CV <= 0 | BOOL |
| **R** | Reset Zählerstand | BOOL | **CV** | Aktueller Zählerstand | INT |
| **LD** | Lade Startwert | BOOL | | | |
| **PV** | Startwert | INT | | | |

Tabelle 61 CTUD Auf- und Abwärtszähler

### 3.5.1.4 SR Flip Flop

Bistabile Funktion, über **S1** wird der Ausgang **Q1** gesetzt und über **R** wieder gelöscht. Liegt an **R** und **S1** zeitgleich ein TRUE an, so ist **S1** dominant.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **S1** | Setzen | BOOL | **Q1** | Ausgang | BOOL |
| **R** | Reset | BOOL | | | |

Tabelle 62 SR Flip Flop

### 3.5.1.5 RS Flip Flop

Bistabile Funktion, über **S** wird der Ausgang **Q1** gesetzt und über **R1** wieder gelöscht. Liegt an **R1** und **S** zeitgleich ein TRUE an, so ist **R1** dominant.

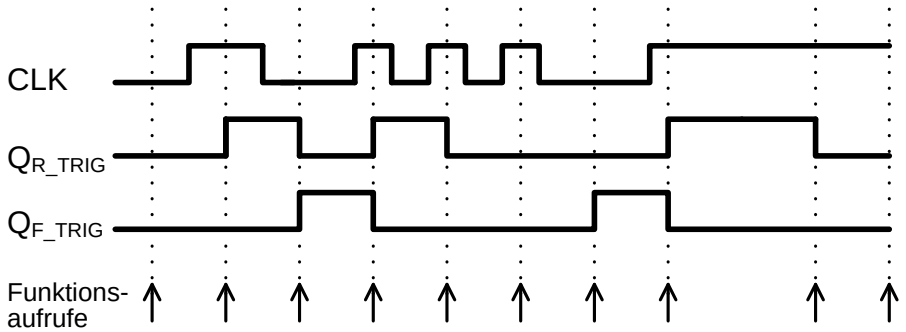
| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **S** | Setzen | BOOL | **Q1** | Ausgang | BOOL |
| **R1** | Reset | BOOL | | | |

Tabelle 63 RS Flip Flop

### 3.5.1.6 R_TRIG und F_TRIG

Beide Funktionen dienen der Flankenerkennung. Wird eine Flanke auf **CLK** erkannt geht **Q** bis zum nächsten Funktionsaufruf auf TRUE, danach wieder auf FALSE. Erst mit einer neuen Flanke kann **Q** wieder für einen Zyklus TRUE werden.

- R_TRIG = steigende Flanke
- F_TRIG = fallende Flanke




| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **CLK** | Setzen | BOOL | **Q** | Ausgang | BOOL |

Tabelle 64 R_TRIG und F_TRIG

**HINWEIS**

Der Ausgang der Funktion ändert sich nur wenn die Funktion auch ausgeführt wird, deshalb ist es ratsam die Flankenerkennung kontinuierlich mit dem SPS Zyklus aufzurufen.

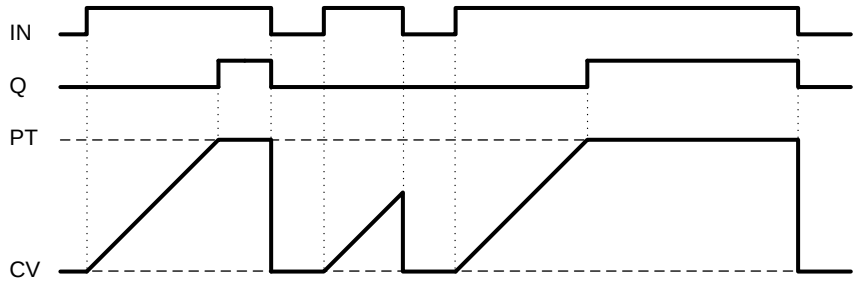
### 3.5.1.7 TON Einschaltverzögerung

Wird **IN** = TRUE gesetzt, dann läuft der Timer hoch. Wenn **ET** = PT ist, wird **Q** auf TRUE gesetzt und der Timer bleibt stehen. **Q** bleibt solange TRUE wie **IN** auch TRUE ist. Bei einer neuen steigenden Flanke auf **IN** fängt der Timer wieder bei null an zu laufen. **PT** kann verändert werden während der Timer läuft.

Die Zeitdauer wird in **PT** in Millisekunden eingegeben. Damit ist eine Zeitverzögerung zwischen 5ms und 24,8 Tagen möglich. Da die Zeitbasis der PLC bei 5ms liegt, ist die minimale Zeitverzögerung auch 5ms.

Für eine vereinfachte Eingabe können hier Literale benutzt werden, wie z.B.

- LD TIME#50s20ms = 50,020 Sekunden
- LD TIME#1d30m = 1 Tag und 30 Minuten




| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **IN** | Timer aktiv | BOOL | **Q** | TRUE ← ( IN=TRUE & ET=PT ) | BOOL |
| **PT** | Zeitdauer | DINT | **ET** | Aktueller Stand des Timers | DINT |

Tabelle 65 TON Einschaltverzögerung

**HINWEIS**

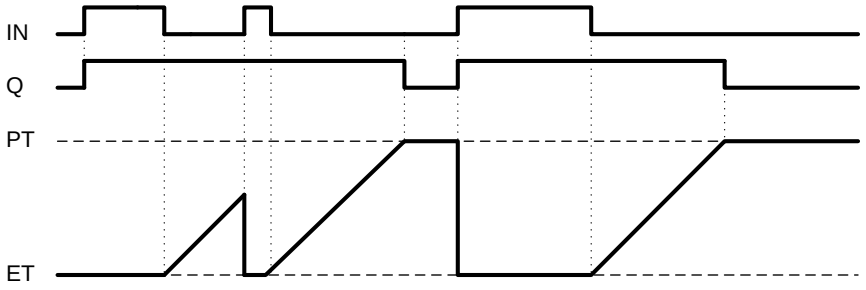
Die Zeit **ET** läuft unabhängig von einem PLC Zyklus. Das Starten des Timers mit **IN** und das Setzen des Ausgangs **Q** werden erst mit dem Funktionsaufruf „CAL“ ausgeführt. Der Funktionsaufruf findet in einem PLC Zyklus statt, dieser kann aber bei längeren PLC Programmen größer 5ms sein, sodass zeitlich ein Jitter entstehen kann.

### 3.5.1.8 TOF Ausschaltverzögerung

Wird **IN** = TRUE, dann wird **Q** auf TRUE gesetzt. Geht **IN** auf FALSE, läuft der Timer hoch. Solange der Timer läuft (**ET** < **PT**) bleibt **Q** auf TRUE gesetzt. Ist (**ET** = **PT**) bleibt der Timer stehen, **Q** wird dann FALSE. Bei einer neuen steigenden Flanke auf **IN**, wird der Timer **ET** wieder auf Null gesetzt.

Für eine vereinfachte Eingabe können hier Literale benutzt werden, wie z.B.

- LD TIME#50s20ms = 50,020 Sekunden
- LD TIME#1d30m = 1 Tag und 30 Minuten




| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **IN** | Timer aktiv | BOOL | **Q** | TRUE ← ( ET<PT ) | BOOL |
| **PT** | Zeitdauer | DINT | **ET** | Aktueller Stand des Timers | DINT |

Tabelle 66 TOF Ausschaltverzögerung

**HINWEIS**

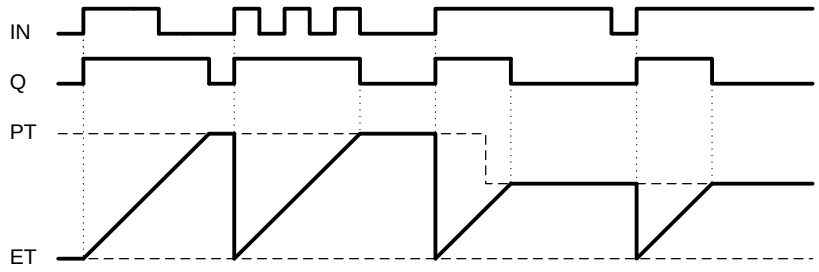
Die Zeit **ET** läuft unabhängig von einem PLC Zyklus. Das Starten des Timers mit **IN** und das Setzen des Ausgangs **Q** werden erst mit dem Funktionsaufruf „CAL“ ausgeführt. Der Funktionsaufruf findet in einem PLC Zyklus statt, dieser kann aber bei längeren PLC Programmen größer 5ms sein, sodass zeitlich ein Jitter entstehen kann.

### 3.5.1.9 TP Zeitimpuls

Bei einer positiven Flanke an **IN** wird der Timer mit dem Wert 0 gestartet. Der Timer zählt bis auf den in **PT** eingetragenen Wert hoch und bleibt dann stehen. Dieser Vorgang ist nicht unterbrechbar! PT kann während des Hochzählens verändert werden. Der Ausgang **Q** ist TRUE, solange der Timer **ET** kleiner als **PT** ist. Wenn **ET = PT** ist und eine steigende Flanke an **IN** erkannt wird, wird der Timer wieder bei 0 gestartet.

Für eine vereinfachte Eingabe können hier Literale benutzt werden, wie z.B.

- LD TIME#50s20ms = 50,020 Sekunden
- LD TIME#1d30m = 1 Tag und 30 Minuten




| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **IN** | Timer aktiv | BOOL | **Q** | TRUE ← ( ET < PT ) | BOOL |
| **PT** | Zeitdauer | DINT | **ET** | Aktueller Stand des Timers | DINT |

Tabelle 67 TP Zeitimpuls

**HINWEIS**

Die Zeit **ET** läuft unabhängig von einem PLC Zyklus. Das Starten des Timers mit **IN** und das Setzen des Ausgangs **Q** werden erst mit dem Funktionsaufruf „CAL“ ausgeführt. Der Funktionsaufruf findet in einem PLC Zyklus statt, dieser kann aber bei längeren PLC Programmen größer 5ms sein, sodass zeitlich ein Jitter entstehen kann.

### 3.5.2 Motion Control

Die Motion Control Lib ist an die PLCopen Spezifikation „Function blocks for motion control“ angelehnt.

Sie enthält Funktionsblöcke (FB) zum Steuern und Verfahren eines Frequenzumrichters und bietet Zugriff auf seine Parameter.

Damit die Motion Blöcke funktionieren, müssen einige Einstellungen in den Parametern des Frequenzumrichters vorgenommen werden.

| Funktionsblock | Benötigte Einstellungen |
|---|---|
| MC_MoveVelocity | • P350 = PLC aktiv<br>• P351 = Hauptsollwert kommt von der PLC<br>• P553 [-xx] = Sollfrequenz<br>• P600 = Lageregelung (Positioniermode) ist ausgeschaltet |
| MC_MoveAbsolute<br>MC_MoveRelative<br>MC_MoveAdditive<br>MC_Home | • P350 = PLC aktiv<br>• P351 = Hauptsollwert kommt von der PLC<br>• P600 = Lageregelung (Positioniermode) ist eingeschaltet<br>• In P553 [-xx] ( PLC_Sollwerte ) muss die Sollposition High Word parametriert sei<br>• In P553 [-xx] ( PLC_Sollwerte ) muss die Sollposition Low Word parametriert sei<br>• In P553 [-xx] ( PLC_Sollwerte ) muss die Sollfrequenz parametriert sei |
| MC_Power<br>MC_Control<br>MC_Reset<br>MC_Stop | • P350 = PLC aktiv<br>• P351 = Steuerwort kommt von der PLC |

Tabelle 68 Einstellungen für die Motion Control FB

**HINWEIS**

Die PLC_Sollwerte 1 bis 5 und das PLC Steuerwort lassen sich auch über Prozessvariablen beschreiben. Sollen jedoch die Motion Control FB‘s verwendet werden, dürfen keine entsprechenden Prozessvariablen in der Variablentabelle deklariert sein, da sonst die Ausgaben der Motion Control FB’s überschrieben werden.

**HINWEIS**

Funktionsblöcke müssen zwingen zyklisch aufgerufen werden! Dies ist wichtig, da die meisten Funktionen in den FB nicht in einem Zyklus abgearbeitet werden.

**HINWEIS**

MC Funktionen, die über einen EXECUTE Eingang gestartet werden, können nicht durch ein Rücksetzen des EXECUTE Einganges gestoppt werden!

| Funktionsblock | Erläuterung |
|---|---|
| MC_ReadParameter | Lesezugriff auf die FU Parameter |
| MC_WriteParameter | Schreibzugriff auf die FU Parameter |
| | |
| MC_MoveVelocity | Verfahrbefehl im Drehzahlmodus |
| MC_MoveAbsolute | Verfahrbefehl mit absoluter Positionsangabe |
| MC_MoveRelative | Verfahrbefehl mit relativer Positionsangabe |
| MC_MoveAdditive | Verfahrbefehl mit additiver Positionsangabe |
| MC_Home | Startet eine Referenzpunktfahrt |
| | |
| MC_Power | Ein-/Ausschalten der Motorspannung |
| MC_Control | Ein-/Ausschalten der Motorspannung + Einstellung der Parametersätze |
| MC_ReadStatus | FU Status |
| MC_ReadActualPos | Liest die aktuelle Position aus |
| MC_Reset | Fehlerreset im FU |
| MC_Stop | Stoppt alle aktiven MC Motionbausteine und schaltet den FU aus |

Tabelle 69 Überblick Motion Control

Die Erreichbarkeit der Umrichterparameter mit Hilfe der Blöcke **MC_ReadParameter** (Kap.: 3.5.2.1) und **MC_WriteParameter** (Kap.: 3.5.2.2) erfolgt über die Parameternummer und den Parameterindex, wobei sich der Parameterindex in Abhängigkeit der Struktur des einzelnen Parameters wie folgt ermittelt:

| Parameternummer | Array des Parameters | Parametersatz | Parameterindex (des Funktionsblocks) |
|---|---|---|---|
| P509 | kein Array- Parameter | Parameter nicht Parame-tersatzabhängig | 0 |
| P510 | [-01] | Parameter nicht Parame-tersatzabhängig | 0 |
| | [-02] | | 1 |
| P104 | kein Array- Parameter | P1 | 0 |
| | | P2 | 1 |
| | | P3 | 2 |
| | | P4 | 3 |
| P613 *(SK53xE)* | [-01] | Parameter nicht Parame-tersatzabhängig | 0 |
| | [-02] | | 1 |
| | [-03] | | 2 |
| | [-04] | | 3 |
| | [-05] | | 4 |
| | [-06] | | 5 |
| P613 *(SK54xE)* | [-01] | P1 | 0 |
| | [-01] | P2 | 1 |
| | [-01] | P3 | 2 |
| | [-01] | P4 | 3 |
| | [-02] | P1 | 4 |
| | [-02] | P2 | 5 |

Tabelle 70 Beispiel Ermittlung Parameterindex des Funktionsblocks

### 3.5.2.1 MC_ReadParameter

Liest einen Parameter zyklisch aus dem Frequenzumrichter, sofern **ENABLE** auf 1 gesetzt ist. Der gelesene Parameter wird in Value abgelegt und ist gültig, wenn **DONE** auf 1 gesetzt ist. Für die Dauer des Lesevorgangs wird der Ausgang **BUSY** 1 auf gesetzt. Bleibt **ENABLE** auf 1 dann wird der Parameter ständig zyklisch ausgelesen. Parameternummer und Index können jederzeit bei aktivem **ENABLE** geändert werden. Jedoch ist schwierig zu erkennen, wann der neue Wert ausgelesen ist, da das **DONE** Signal die gesamte Zeit 1 ist. In diesem Fall ist es empfehlenswert das **ENABLE** Signal für einen Zyklus auf 0 zu setzen, da das **DONE** Signal dann zurückgesetzt wird. Der Parameterindex ergibt sich aus dem Index in der Dokumentation minus 1 (Siehe Tabelle 70). So wird z.B. P700 Index 3 („Grund Einschaltsperre“) über den Parameterindex 2 abgefragt. Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0 und die **ERRORID** enthält den Fehlercode. Wird das **ENABLE** Signal auf 0 gesetzt, dann werden alle Signale und die **ERRORID** gelöscht.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Freigabe | BOOL | **DONE** | Value ist gültig | BOOL |
| **PARAMETERNUMBER** | Parameternummer | INT | **BUSY** | Der Vorgang ist nicht abgeschlossen | BOOL |
| **PARAMETERINDEX** | Parameterindex | INT | **ERROR** | Lesevorgang ist fehlgeschlagen | BOOL |
| | | | **ERRORID** | Fehlercode | INT |
| | | | **VALUE** | Ausgelesener Parameter | DINT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | unzulässige Parameternummer |
| 3 | fehlerhafter Parameterindex |
| 4 | kein Array |
| 201 | Ungültiges Auftragselement im zuletzt empfangenen Auftrag |
| 202 | Interne Antwortkennung nicht abbildbar |

Tabelle 71 MC_ReadParameter

### 3.5.2.2 MC_WriteParameter_16 / MC_WriteParameter_32

Schreibt einen 16/32 Bit Parameter in den Frequenzumrichter, wenn **EXECUTE** von 0 auf 1 wechselt (Flanke). Der Parameter wurde geschrieben, wenn **DONE** auf 1 gesetzt ist. Für die Dauer des Lesevorgangs wird der Ausgang **BUSY** 1 auf  gesetzt. Im Fehlerfall wird **ERROR** auf 1 gesetzt und die **ERRORID** enthält den Fehlercode. Die Signale **DONE**, **ERROR**, **ERRORID** bleiben solange gesetzt, bis **EXECUTE** wieder auf 0 wechselt. Wechselt das **EXECUTE** Signal auf 0, dann wird der Schreibprozess nicht abgebrochen. Nur das **DONE** Signal bleibt nur für 1 PLC Zyklus gesetzt.
Wird der Eingang **RAMONLY** auf 1 gesetzt, dann wird nur RAM gespeichert. Mit einem Ausschalten des FU gehen die so veränderten Einstellungen verloren.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Freigabe | BOOL | **DONE** | Value ist gültig | BOOL |
| **PARAMETERNUMBER** | Parameternummer | INT | **BUSY** | Der Schreibvorgang ist aktiv | BOOL |
| **PARAMETERINDEX** | Parameterindex | INT | **ERROR** | Lesevorgang ist fehlgeschlagen | BOOL |
| **VALUE** | Zu schreibender Wert | INT, DINT | **ERRORID** | Fehlercode | INT |
| **RAMONLY** | Speichern nur im RAM | | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | unzulässige Parameternummer |
| 1 | Parameterwert nicht änderbar |
| 2 | untere oder obere Wertgrenze überschritten |
| 3 | fehlerhafter Parameterindex |
| 4 | kein Array |
| 5 | Unzulässiger Datentyp |
| 6 | Nur rücksetzbar (es darf nur 0 geschrieben werden) |
| 7 | Beschreibungselement nicht änderbar |
| 201 | Ungültiges Auftragselement im zuletzt empfangenen Auftrag |
| 202 | Interne Antwortkennung nicht abbildbar |

Tabelle 72 MC_WriteParameter16 / MC_WriteParameter32

### 3.5.2.3 MC_MoveVelocity

Setzt die Sollfrequenz für den Frequenzumrichter, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Hat der Frequenzumrichter die Sollfrequenz erreicht, so wird **INVELOCITY** auf 1 gesetzt. Während der FU auf die Sollfrequenz beschleunig, ist der **BUSY** Ausgang aktiv. Wurde **EXECUTE** bereits auf 0 gesetzt, dann wird **INVELOCITY** nur für einen Zyklus auf 1 gesetzt.

Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt.

Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

**VELOCITY** wird skaliert nach folgender Formel eingegeben:

**VELOCITY** = ( Sollfrequenz (Hz) × 0x4000 ) / P105

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Freigabe | BOOL | **INVELOCITY** | Vorgegebene Sollfrequenz erreicht | BOOL |
| **VELOCITY** | Sollfrequenz | INT | **BUSY** | Sollfrequenz noch nicht erreicht | BOOL |
| | | | **COMMAND-ABORTED** | Befehl abgebrochen | BOOL |
| | | | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1000h | FU ist nicht freigegeben |
| 1100h | FU nicht im Drehzahl Mode |
| 1101h | Keine Sollfrequenz parametriert (P553) |

Tabelle 73 MC_MoveVelocity

### 3.5.2.4 MC_MoveAbsolute

Schreibt einen Positions- und Geschwindigkeitssollwert zum Frequenzumrichter, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Die Sollfrequenz **VELOCITY** wird nach der im MC_MoveVelocity erläuterten Skalierung übergeben.

Das High und das Low Word der Sollposition sind im Parameter P553 zu definieren.

**POSITION:**

**MODE** = False:

Die Sollposition ergibt sich aus dem in **POSITION** übergebenen Wert.

**MODE** = True:

Der in **POSITION** übergebene Wert entspricht dem um 1 erhöhten Index aus Parameter P613. Die in diesem Parameterindex hinterlegte Position entspricht der Sollposition.

Beispiel:

Mode = True; Position = 12

→ Der FB fährt die Position, die im aktuellen Parametersatz von P613[-13] steht, an.

Hat der Umrichter die Sollposition erreicht, so wird **DONE** auf 1 gesetzt. **DONE** wird mit dem Rücksetzen von **EXECUTE** gelöscht. Wenn **EXECUTE** vor dem Erreichen der Zielposition gelöscht wird, so wird **DONE** für einen Zyklus auf 1 gesetzt. Während des Verfahrens zur Sollposition ist **BUSY** aktiv.

Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt.

Im Fehlerfall wird **ERROR** auf 1 und in **ERRORID** der entsprechende Fehlercode gesetzt. **DONE** ist in diesem Fall 0.

Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Freigabe | BOOL | **DONE** | Vorgegebene Sollposition erreicht | BOOL |
| **POSITION** | Sollposition | DINT | **BUSY** | Sollposition nicht erreicht | BOOL |
| **VELOCITY** | Sollfrequenz | INT | **COMMAND-ABORTED** | Befehl abgebrochen | BOOL |
| **MODE** | Modus Quelle Sollposition | BOOL | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 0x1000 | FU ist nicht freigegeben |
| 0x1200 | Lageregelung ist nicht aktiviert |
| 0x1201 | In den PLC Sollwerten ist die High Position nicht eingetragen (P553) |
| 0x1202 | In den PLC Sollwerten ist die Low Position nicht eingetragen (P553) |

Tabelle 74 MC_MoveAbsolute

### 3.5.2.5 MC_MoveRelative

Entspricht bis auf den Eingang **DISTANCE** in allen Punkten dem MC_MoveAbsolute. Die Sollposition ergibt sich aus der Addition von der aktuellen Istposition und der übergebenen **DISTANCE**.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Freigabe | BOOL | **DONE** | Vorgegebene Sollposition erreicht | BOOL |
| **DISTANCE** | Sollposition | DINT | **BUSY** | Sollposition nicht erreicht | BOOL |
| **VELOCITY** | Sollfrequenz | INT | **COMMAND-ABORTED** | Befehl abgebrochen | BOOL |
| **MODE** | Modus Quelle Sollposition | BOOL | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1000h | FU ist nicht freigegeben |
| 1200h | Lageregelung ist nicht aktiviert |
| 1201h | In den PLC Sollwerten ist die High Position nicht eingetragen (P553) |
| 1202h | In den PLC Sollwerten ist die Low Position nicht eingetragen (P553) |

Tabelle 75 MC_MoveRelative

### 3.5.2.6 MC_MoveAdditive

Entspricht bis auf den Eingang **DISTANCE** in allen Punkten dem MC_MoveAbsolute. Die Sollposition ergibt sich aus der Addition von aktueller Sollposition und der übergebenen **DISTANCE**.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Freigabe | BOOL | **DONE** | Vorgegebene Sollposition erreicht | BOOL |
| **DISTANCE** | Sollposition | DINT | **BUSY** | Sollposition nicht erreicht | BOOL |
| **VELOCITY** | Sollfrequenz | INT | **COMMAND-ABORTED** | Befehl abgebrochen | BOOL |
| **MODE** | Modus Quelle Sollposition | BOOL | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1000h | FU ist nicht freigegeben |
| 1200h | Lageregelung ist nicht aktiviert |
| 1201h | In den PLC Sollwerten ist die High Position nicht eingetragen (P553) |
| 1202h | In den PLC Sollwerten ist die Low Position nicht eingetragen (P553) |

Tabelle 76 MC_MoveAdditive

### 3.5.2.7 MC_Home

Veranlasst den Frequenzumrichter eine Referenzpunktfahrt zu starten, sofern **EXECUTE** von 0 auf 1 wechselt (Flanke). Der Frequenzumrichter verfährt mit der in **VELOCITY** eingetragenen Sollfrequenz. Wenn der Eingang mit dem Positionsreferenzsignal (P420[-xx] = Referenzpunkt) aktiv wird, dann erfolgt eine Drehrichtungsumkehr. Bei der negativen Flanke des Positionsreferenzsignals wird der in **POSITION** stehende Wert übernommen. Anschließend bremst der Frequenzumrichter auf 0Hz ab, das Signal **DONE** geht auf 1. Während der gesamten HOME Fahrt ist der **BUSY** Ausgang aktiv.

Wird der Eingang MODE auf 1 gesetzt so bleibt der FU nach einer Homefahrt auf der Initiator-Mitte stehen.

Sollte der Vorgang abgebrochen werden (z.B. durch einen anderen MC Funktionsbaustein), wird **COMMANDABORTED** gesetzt.

Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Freigabe | BOOL | **DONE** | Vorgegebene Sollposition erreicht | BOOL |
| **POSITION** | Sollposition | DINT | **BUSY** | Home Fahrt aktiv | BOOL |
| **VELOCITY** | Sollfrequenz | INT | **COMMAND-ABORTED** | Befehl abgebrochen | BOOL |
| **MODE** | Home Mode | BOOL | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1000h | FU ist nicht freigegeben |
| 1200h | Lageregelung ist nicht aktiviert |
| 1201h | In den PLC Sollwerten ist die High Position nicht eingetragen (P553) |
| 1202h | In den PLC Sollwerten ist die Low Position nicht eingetragen (P553) |

Tabelle 77 MC_Home

### 3.5.2.8 MC_Power

Über diese Funktion kann die Endstufe des Frequenzumrichters ein- oder ausgeschaltet werden. Wird der **ENABLE** Eingang auf 1 gesetzt, dann wird die Endstufe freigegeben. Voraussetzung dafür ist, dass sich der FU im Status „Einschaltsperre“ oder „Einschaltbereit“ befindet. Sollte der FU im Status „Störung“ oder „Störungsreaktion aktiv“ sein, muss zuerst die Störung beseitig und quittiert werden. Erst dann kann eine Freigabe über diesen Block erfolgen. Befindet sich der Frequenzumrichter im Status „Nicht Einschaltbereit“, ist ein Einschalten auch nicht möglich. In allen Fällen geht der FB in den Fehlerstatus und **ENABLE** muss auf 0 gesetzt werden, um den Fehler zu quittieren.

Wird der **ENABLE** Eingang auf 0 gesetzt, dann wird der Frequenzumrichter ausgeschaltet. Geschieht dies bei laufendem Motor, so wird dieser über die in P103 eingestellte Rampe vorher auf 0Hz heruntergefahren.

Der **STATUS** ist 1, wenn die Endstufe des Frequenzumrichters eingeschaltet ist, andernfalls ist er 0.

**ERROR** und **ERRORID** werden zurückgesetzt, wenn **ENABLE** auf 0 geschaltet wird.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Freigabe | BOOL | **STATUS** | Motor wird bestromt | BOOL |
| | | | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1001h | Stop Funktion ist aktiv |
| 1300h | FU befindet sich nicht im Status „Einschaltbereit“ oder „Einschaltsperre“ |

Tabelle 78 MC_Power

### 3.5.2.9 MC_Control

Dieser FB dient zum Steuern des FU und bildet die Möglichkeiten des FU Steuerwortes etwas detaillierter nach als der MC_Power. Über die Eingänge **ENABLE**, **DISABLEVOLTAGE** und **QUICKSTOP** wird der FU gesteuert, siehe nachfolgende Tabelle.

| Baustein Eingänge | | | Verhalten Frequenzumrichter |
|---|---|---|---|
| **ENABLE** | **QUICKSTOP** | **DISABLEVOLTAGE** | |
| High | Low | Low | Der Frequenzumrichter wird eingeschaltet. |
| Low | Low | Low | Der Frequenzumrichter bremst auf 0Hz (P103) und schaltet dann den Motor spannungsfrei. |
| × | × | High | Der Frequenzumrichter wird sofort spannungsfrei geschaltet, der Motor dreht ungebremst aus. |
| × | High | Low | Der Frequenzumrichter fährt einen Schnellstopp (P426) und schaltet dann den Motor spannungsfrei |

Tabelle 79 Belegung der Steuereingänge ( × = Pegel am Eingang ist unwichtig)

Über den Eingang **PARASET** kann der aktive Parametersatz eingestellt werden.

Wenn der Ausgang **STATUS** = 1 ist, dann ist der FU eingeschaltet und der Motor wird bestromt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Freigabe | BOOL | **STATUS** | Motor wird bestromt | BOOL |
| **DISABLEVOLTAGE** | Sannungsfreischalten | BOOL | **ERROR** | Fehler im FB | BOOL |
| **QUICKSTOP** | Schnellstop | BOOL | **ERRORID** | Fehlercode | INT |
| **PARASET** | Aktiver Parametersatz<br>Wertebereich: 0 - 3 | BYTE | | | |
| **ERRORID** | **Erläuterung** | | | | |
| 0 | Kein Fehler | | | | |
| 1001h | Stop Funktion ist aktiv | | | | |
| 1300h | FU befindet sich in einem unerwarteten State | | | | |

Tabelle 80 MC_Control

### 3.5.2.10 MC_ReadStatus

Liest den Status des Frequenzumrichters aus. Die Statusmaschine orientiert sich an der PLCopen Spezifikation „Function blocks for motion control“. Solange **ENABLE** auf 1 steht wird der Zustand ausgelesen.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Freigabe | BOOL | **VALID** | Ausgang ist gültig | BOOL |
| | | | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORSTOP** | FU hat einen Fehler | BOOL |
| | | | **DISABLED** | FU Endstufe ist ausgeschaltet | BOOL |
| | | | **STOPPING** | Ein Stop Befehl ist aktiv | BOOL |
| | | | **DISCRETEMOTION** | Einer der drei Positionier FB ist aktiv | BOOL |
| | | | **CONTINUOUSMOTION** | Der MC_Velocity ist aktiv | BOOL |
| | | | **HOMING** | Der MC_Home ist aktiv | BOOL |
| | | | **STANDSTILL** | Der FU hat keinen aktiven Verfahrbefehl. Er steht mit Drehzahl 0U/min und eingeschalteter Endstufe. | BOOL |

Tabelle 81 MC_ReadStatus

### 3.5.2.11 MC_ReadActualPos

Liefert kontinuierlich die aktuelle Istposition des Frequenzumrichters, wenn **ENABLE** auf 1 steht. Sobald eine gültige Istposition am Ausgang anliegt wird **VALID** auf gültig gesetzt. Im Fehlerfall wird ERROR auf 1 gesetzt und VALID ist in diesem Fall 0.

Skalierung Position: 1 Motorumdrehung = 1000

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Freigabe | BOOL | **VALID** | Ausgang ist gültig | BOOL |
| | | | **ERROR** | Fehler im FB | BOOL |
| | | | **POSITION** | Aktuelle Istposition des FU | DINT |

Tabelle 82 MC:ReadActualPos

### 3.5.2.12 MC_Reset

Rücksetzen eines Fehlers im Frequenzumrichter (Störungsquittierung), bei einer steigenden Flanke von **EXECUTE**. Im Fehlerfall wird **ERROR** auf 1 gesetzt und die Fehlerursache in **ERRORID** eingetragen. Bei einer negativen Flanke an **EXECUTE** werden alle Fehler zurückgesetzt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Start | BOOL | **DONE** | FU Fehler zurückgesetzt | BOOL |
| | | | **BUSY** | Resetvorgang ist noch aktiv | BOOL |
| | | | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1001h | Stop Funktion ist aktiv |
| 1700h | Ein Fehler – Reset konnte nicht ausgeführt werden, die Ursache für den Fehler liegt noch an |

Tabelle 83 MC_Reset

### 3.5.2.13 MC_Stop

Bei steigender Flanke (0 auf 1) wird der Frequenzumrichter in den Zustand **STANDINGSTILL** gesetzt. Alle gerade aktiven Motion Funktionen werden abgebrochen. Der Frequenzumrichter bremst auf 0Hz ab und schaltet die Endstufe aus. Solange der Stop Befehl aktiv ist (**EXECUTE** = 1), werden alle anderen Motion FB geblockt. Der **BUSY** Ausgang wird mit der steigenden Flanke an **EXECUTE** aktiv und bleibt dies solange bis eine fallende Flanke an **EXECUTE** erfolgt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Start | BOOL | **DONE** | Befehl ist ausgeführt | BOOL |
| | | | **BUSY** | Befehl ist aktiv | BOOL |

Tabelle 84 MC_Stop

### 3.5.3 Elektronisches Getriebe mit „fliegender Säge“

Für das *elektronische Getriebe* („winkelsynchroner Gleichlauf“) und die Unterfunktion *Fliegende Säge* gibt es zwei Funktionsblöcke, die eine Steuerung dieser Funktionen erlauben. Weiterhin müssen für einen korrekten Ablauf der beiden Funktionsblöcke im Master- und Slave- Frequenzumrichter diverse Parameter eingestellt werden. Exemplarisch ist dies in der nachfolgenden Tabelle aufgeführt.

| **Master FU** | | | **Slave FU** | | |
|---|---|---|---|---|---|
| Parameter | Einstellung | Bedeutung | Parameter | Einstellung | Bedeutung |
| P502[01] | 20 | Sollfreq. nach Freq.Rampe | P509 | 10 * | CANopen Broadcast * |
| P502[02] | 15 | Istpos in Inc. High – Word | P510[-01] | 10 | CANopen Broadcast |
| P502[-03] | 10 | Istpos in Inc. Low – Word | P510[-02] | 10 | CANopen Broadcast |
| P503 | 3 | CANopen | P505 | 0 | 0,0Hz |
| P505 | 0 | 0,0Hz | P515[-02] | $P515[-03]_{Master}$ | Broadcast Slave Adresse |
| P514 | 5 | 250kBaud (min. 100kBaud) | P546[-01] | 4 | Frequenzaddition |
| P515[-03] | $P515[-02]_{Slave}$ | Broadcast Master Adresse | P546[-02] | 24 | Sollpos. Inc. High – Word |
| | | | P546[-03] | 23 | Sollpos. Inc. Low – Word |
| | | | P600 | 1,2 | Lageregelung an |
| | | | | | |
| | | | Nur für den FB_Gearing | | |
| | | | P553[-01] | 21 | Pos. Sollpos Low Word |
| | | | P553[-02] | 22 | Pos. Sollpos High Word |
| * (P509) muss nicht zwingend auf {10} „CANopen Broadcast“ stehen. Dann jedoch ist am Master (P502 [-01]) auf die Einstellung {21} "Istfrequenz ohne Schlupf" zu stellen. | | | | | |

Tabelle 85 Parametrierung für Getriebefunktion

**HINWEIS**

Die Istlage des Master muss zwingend im Format „Inkremente“ (Inc) übergeben werden.

#### 3.5.3.1 Überblick

| **Funktionsbaustein** | **Erläuterung** |
|---|---|
| FB_Gearing | FB für die einfache Getriebefunktion |
| FB_FlyingSaw | FB für Getriebefunktion mit fliegender Säge |

Tabelle 86 Überblick Elektronisches Getriebe

### 3.5.3.2 FB_Gearing

Über den Funktionsbaustein FB_Gearing kann die Position und die Drehzahl des Frequenzumrichters auf die eines Masterumrichters synchronisiert werden. Der Slave, der diese Funktion verwendet, folgt immer den Bewegungen des Masterumrichters.

Die Synchronisierung erfolgt absolut, d.h. Slave- und Masterposition sind immer gleich.

**HINWEIS**

Wird der Slave mit einer anderen Position als der Master in den Getriebemodus geschaltet, dann verfährt der Slave mit max. Frequenz zur Masterposition.

Wird ein Übersetzungsverhältnis angegeben, ergibt sich nach dem Wiedereinschalten auch eine neue Position.

Der Positionswert, auf den synchronisiert wird, sowie die Drehzahl, müssen über den Broadcast Kanal übertragen werden.

Über den Eingang **ENABLE** wird die Funktion aktiviert, dabei muss die Lagereglung aktiv und die Endstufe freigegeben sein. Die Endstufe kann z.B. mit der Funktion MC_Power freigegeben werden. Wird **ENABLE** auf 0 gesetzt, dann bremst der Frequenzumrichter auf 0Hz und bleibt stehen. Der Umrichter befindet sich jetzt wieder im Modus Lageregelung.

Wird der MC_Stop aktiviert, dann verlässt der Frequenzumrichter den Getriebemodus und der **ABORT** Ausgang geht auf 1. Bei Fehlern im FB geht **ERROR** auf 1 und die Fehlerursache steht in **ERRORID.** Über ein setzten von **ENABLE** auf 0 kann **ERROR, ERRORID** und **ABORT** wieder zurückgesetzt werden.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Gleichlauf aktiv | BOOL | **VALID** | Getriebefunktion ist aktiv | BOOL |
| | | | **ABORT** | Befehl abgebrochen | BOOL |
| | | | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1000h | FU ist nicht freigegeben |
| 1200h | Lageregelung ist nicht aktiviert |
| 1201h | Der PLC Sollwert Position High ist nicht parametriert |
| 1202h | Der PLC Sollwert Position Low ist nicht parametriert |

Tabelle 87 FB_Gearing

Eine Über-/Untersetzung bzw. ein Drehrichtungswechsel kann durch die Parameter P607[-05] bzw. P608[-05] eingestellt werden. Näheres dazu ist im Handbuch BU0510 (Zusatzanleitung für die Positioniersteuerung POSICON) zu lesen.

### 3.5.3.3 FB_FlyingSaw

Die Funktion Fliegende Säge stellt eine Erweiterung zur Getriebefunktion dar. Mit Hilfe dieser Funktion ist es möglich auf einen fahrenden Antrieb positionsgenau zu synchronisieren. Die Synchronisierung erfolgt im Gegensatz zu FB_Gearing relativ, d.h. die Slave Achse verfährt synchron zu der Position des Masters, die beim Start der „Fliegenden Säge“ anlag. Der Vorgang der Synchronisierung ist im nachfolgenden Bild dargestellt.

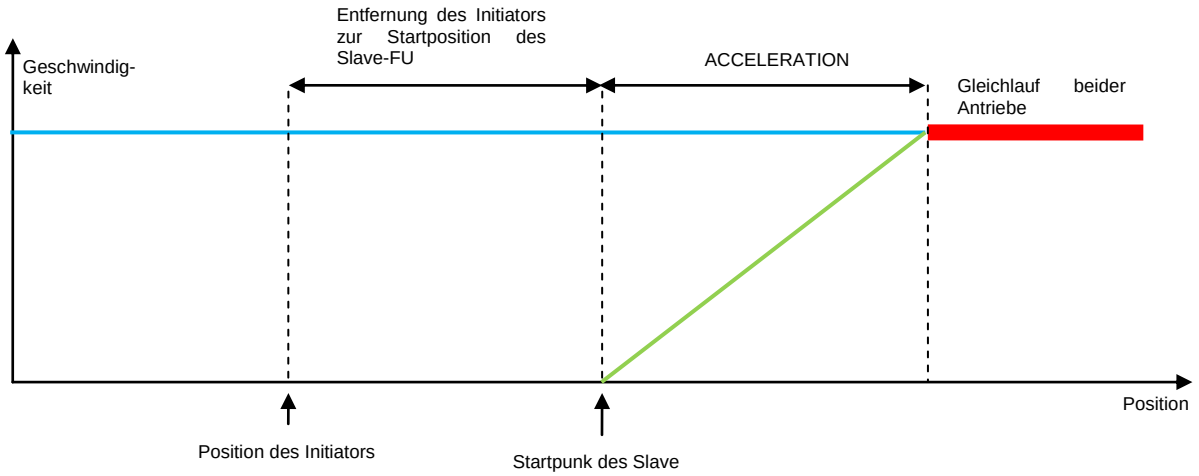



Abbildung 7 Prinzipdarstellung Synchronisationsablauf Fliegenden Säge

Wird die Funktion gestartet, dann beschleunigt der Slave Frequenzumrichter auf die Geschwindigkeit der Masterachse. Die Beschleunigungsrampe wird über den „Beschleunigungsweg“ **ACCELERATION** festgelegt. Bei niedrigen Geschwindigkeit ist die Rampe so flacher und bei hohen Master Geschwindigkeiten ergibt sich eine steilere Rampe für den Slave Frequenzumrichter. Der Beschleunigungsweg wird in Umdrehungen (1000 = 1,000 rev) angegeben, wenn P553 als Sollposition angegeben ist. Wird für P553 Sollposition INC verwendet, dann wird der Beschleunigungsweg in Inkrementen angegeben.

Wird der Initiator mit der in **ACCELERATION** gespeicherten Entfernung vor die Position des Slave Antriebes gesetzt, dann wird der Slave präzise mit der auslösenden Position auf dem Masterantrieb synchronisiert.

Der FB muss über den **ENABLE** Eingang eingeschaltet werden. Der Start der Funktion kann entweder über einen digitalen Eingang (P420[-xx]=64, *Start Fliegende Säge*) oder **EXECUTE** erfolgen. Der Frequenzumrichter beschleunigt dann auf die Geschwindigkeit der Masterachse. Bei Erreichen der Synchronität zur Masterachse wird der **DONE** Ausgang auf 1 geschaltet.

Über den **STOP** Eingang oder die digitale Eingangsfunktion P420[-xx] = 77, *Fliegende Säge anhalten*, erfolgt ein Ausschalten der Getriebefunktion, der Frequenzumrichter bremst auf 0Hz und bleibt stehen. Über den **HOME** Eingang wird der Umrichter veranlasst auf die absolute Position 0 zu fahren. Nach Beendigung des **HOME** oder **STOP** Befehls ist der jeweils zugeordnete Ausgang aktiv. Über eine erneute Betätigung von **EXECUTE** oder den digitalen Eingang kann die Getriebefunktion wieder gestartet werden. Mit der digitalen Eingangsfunktion (P420[-xx] = 63, *Gleichlauf ausschalten*) kann die Getriebefunktion angehalten, und anschließend auf die absolute Position 0 gefahren werden.

Wird die Funktion durch die MC_Stop Funktion unterbrochen, dann wird **ABORT** auf 1 gesetzt. Im Fehlerfall wird **ERROR** auf 1 und in **ERRORID** der Fehlercode gesetzt. Diese drei Ausgänge werden zurückgesetzt wenn **ENABLE** auf 0 geschaltet wird.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Freigabe | BOOL | **VALID** | Vorgegebene Sollfrequenz erreicht | BOOL |
| **EXECUTE** | Start der Synchronisierung | BOOL | **DONEHOME** | Home Fahrt beendet | |
| **STOP** | Stop der Synchronisierung | BOOL | **DONESTOP** | Stop Kommando ausgeführt | |
| **HOME** | Verfährt auf Position 0 | BOOL | **ABORT** | Befehl abgebrochen | BOOL |
| **ACCELERATION** | Beschleunigungsweg (1rev. = 1.000) | DINT | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1000h | FU ist nicht freigegeben |
| 1200h | Lageregelung ist nicht aktiviert |

Tabelle 88 FB_FlyingSaw

### 3.5.4 FB_FunctionCurve

Der Funktionsbaustein stellt eine Kennfeldsteuerung dar. Es können dem Funktionsblock definierte Punkte übergeben werden, durch die er eine Funktion emuliert. Der Ausgang verhält sich dann entsprechend des hinterlegten Kennfelds. Zwischen den einzelnen Stützpunkten wird linear Interpoliert.

Die Stützstellen werden mit X und Y-Werten definiert. Die X-Werte sind dabei immer vom Typ **INT**, die Y-Werte können alle entweder vom Typ **INT** oder **DINT** sein, je nach Größe der größten Stützstelle. Wird **DINT** verwendet verbraucht dies auch mehr Speicherplatz.

Die Stützstellen werden im Variablenfenster in der Spalte „Init-Wert“ eingetragen.

Wird am Eingang **ENABLE** ein TRUE erkannt wurde, wird anhand des Eingangswerts **INVALUE** der entsprechende Ausgangswert **OUTVALUE** berechnet. **VALID** signalisiert mit einem TRUE, dass der Ausgangwert **OUTVALUE** gültig ist.

Solange **VALID** FALSE ist, hat der Ausgang **OUTVALUE** den Wert 0.

Überschreitet der Eingangswert **INVALUE** das obere oder untere Ende des Kennfeldes, dann bleibt der erste oder letzte Ausgangswert des Kennfeldes am Ausgang solange stehen, bis sich **INVALUE** wieder im Bereich des Kennfeldes befindet. Bei Über- oder Unterschreitung des Kennfeldes wird der entsprechende Ausgang **MINLIMIT** oder **MAXLIMIT** auf TRUE gesetzt.

**ERROR** wird TRUE, wenn die Abszissenwerte (X-Werte) des Kennfeldes nicht fortlaufen größer werden, oder keine Tabelle initialisiert wird. Dabei wird der entsprechende Fehler auch über **ERRORID** ausgegeben und der Ausgangswert wird 0. Der Fehler wird zurückgesetzt, wenn **ENABLE** = FALSE wird.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Ausführen | BOOL | **VALID** | Ausgangswert ist gültig | BOOL |
| **INVALUE** | Eingangswert ( x ) | INT | **ERROR** | Fehler im FB | BOOL |
| | | | **ERRORID** | Fehlercode | INT |
| | | | **MAXLIMIT** | Maximales Limit erreicht | BOOL |
| | | | **MINLIMIT** | Minimales Limit erreicht | BOOL |
| | | | **OUTVALUE** | Ausgangswert ( y ) | DINT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1400h | Abszissenwerte (X-Werte) des Kennfeldes nicht immer steigend |
| 1401h | Kein Kennfeld initialisiert |

Tabelle 89 FB_FunctionCurve

### 3.5.5 FB_PIDT1

Der P-I-DT1 stellt einen frei parametrierbaren diskreten Regler dar. Werden einzelne Anteile nicht benötigt, sowie der P, der I oder der DT1 Anteil, wird dessen Parameter mit 0 beschrieben. Der T1 Anteil arbeitet nur mit dem D Anteil zusammen. Es lässt sich also kein PT1 Regler parametrieren. Auf Grund von interner Speicherbegrenzung sind die Regelungsparameter auf folgende Bereiche begrenzt:

| Zulässiger Wertebereich für Regelungsparameter | | | |
|---|---|---|---|
| **Parameter** | **Wertebereich** | **Skalierung** | **resultierender Wertebereich** |
| **P (Kp)** | 0 – 32767 | 1/100 | 0,00 – 32,767 |
| **I (Ki)** | 0 – 10240 | 1/100 | 0,00 – 10,240 |
| **D (Kd)** | 0 – 32767 | 1/1000 | 0,000 – 3,2767 |
| **T1 (ms)** | 0 – 32767 | 1/1000 | 0,000 – 3,2767 |
| **Max** | -32768 – 32767 | | |
| **Min** | -32768 – 32767 | | |

Tabelle 90 Wertebereich Regelungsparameter FB_PIDT1

Wenn der Eingang **ENABLE** auf TRUE gesetzt wird, beginnt der Regler zu rechnen. Die Regelungsparameter werden nur bei der steigenden Flanke von **ENABLE** übernommen. Während **ENABLE** auf TRUE ist, bleibt ein Verändern der Reglungsparameter wirkungslos. Wird **ENABLE** auf FALSE gesetzt, bleibt der Ausgang auf dem letzten Wert stehen.

Das Ausgangsbit **VALID** wird gesetzt, solange sich der Ausgangswert Q innerhalb der Grenzen Min und Max bewegt und der Eingang **ENABLE** auf TRUE steht.

**ERROR** wird gesetzt, sobald ein Fehler aufgetreten ist. Das Bit **VALID** ist dann FALSE und die Fehlerursache ist über **ERRORID** (siehe Tabelle unten) zu erkennen.

Wird das Bit **RESET** auf TRUE gesetzt, werden der Integrator- und der Differenziatorinhalt auf 0 gesetzt. Ist der Eingang **ENABLE** auf FALSE, wird auch der Ausgang **OUTPUT** auf 0 gesetzt. Ist der Eingang **ENABLE** auf TRUE gesetzt, wirkt somit nur der P-Anteil auf den Ausgang **OUTPUT**.

Überschreitet der Ausgangswert **OUTPUT** die maximalen oder minimalen Ausgangswerte, wird das entsprechende Bit **MAXLIMIT** bzw. **MINLIMIT** gesetzt und das Bit **VALID** wird auf FALSE gesetzt.

**HINWEIS**

Kann das gesamte Programm nicht innerhalb von einem PLC Zyklus abgearbeitet werden, rechnet der Regler den Ausgangswert ein zweites Mal mit den alten Abtastwerten. Dadurch wird eine konstante Abtastrate erreicht. Aus diesem Grund ist es notwendig, dass der CAL Befehl für den PIDT1 Regler in jedem PLC Zyklus und nur am Ende des PLC Programms ausgeführt wird!

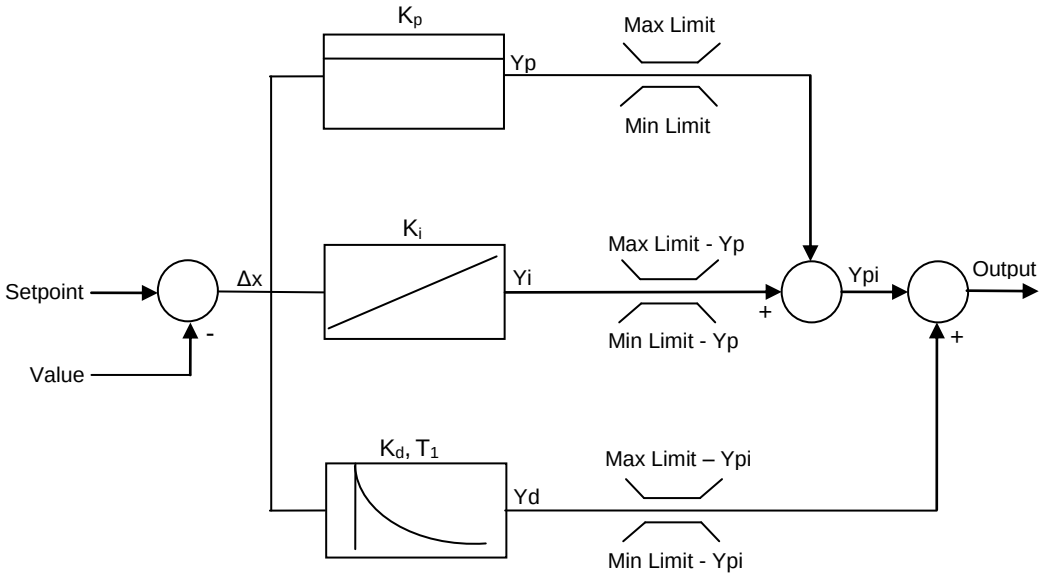



Abbildung 8 Sollwertverarbeitung FB PIDT1

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Ausführen | BOOL | **VALID** | Ausgangswert ist gültig | BOOL |
| **RESET** | Ausgangswerte zurücksetzen | BOOL | **ERROR** | Fehler im FB | BOOL |
| **P** | P-Anteil (Kp) | INT | **ERRORID** | Fehlercode | INT |
| **I** | I-Anteil (Ki) | INT | **MAXLIMIT** | Maximales Limit erreicht | BOOL |
| **D** | D-Anteil (Kd) | INT | **MINLIMIT** | Minimales Limit erreicht | BOOL |
| **T1** | T1-Anteil in ms | INT | **OUTPUT** | Ausgangswert | INT |
| **MAX** | Maximaler Ausgangswert | INT | | | |
| **MIN** | Minimaler Ausgangswert | INT | | | |
| **SETPOINT** | Sollwert | INT | | | |
| **VALUE** | Istwert | INT | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1600h | P-Anteil nicht im Wertebereich |
| 1601h | I-Anteil nicht im Wertebereich |
| 1602h | D-Anteil nicht im Wertebereich |
| 1603h | T1-Anteil nicht im Wertebereich |

Tabelle 91 FB_PIDT1

## 3.5.6 Visualisierung ParameterBox

In der ParameterBox kann der komplette Displayinhalt für eigene Informationsdarstellungen benutzt werden. Dazu muss die ParameterBox in den Visualisierungsmodus geschaltet werden. Dies ist ab der Firmware Version V4.3 der ParameterBox (Parameter P1308) möglich und geschieht wie folgt:

- Im Menüpunkt „Anzeige“ den Parameter P1003 auf „PLC-Anzeige“ einstellen
- Über die rechte oder linke Pfeiltaste auf die Betriebswertanzeige wechseln
- PLC Anzeige ist jetzt in der P-Box aktiv und bleibt dies auch dauerhaft

Im Visualisierungsmodus der P-Box kann über die zwei nachfolgend erläuterten FB der Displayinhalt beschrieben werden. Vorab muss jedoch im PLC Konfigurationsdialog (Schaltfläche ), der Punkt „Parameterbox Funktionsbausteine zulassen“ aktiviert sein.

Über den Prozesswert „Parameterbox_key_state“ kann zusätzlich der Tastaturzustand der Box abgefragt werden. Damit können Eingaben in das PLC Programm realisiert werden.

Der nachfolgenden Abbildung kann der Displayaufbau und die Position der auszulesenden Tasten für die ParameterBox entnommen werden.

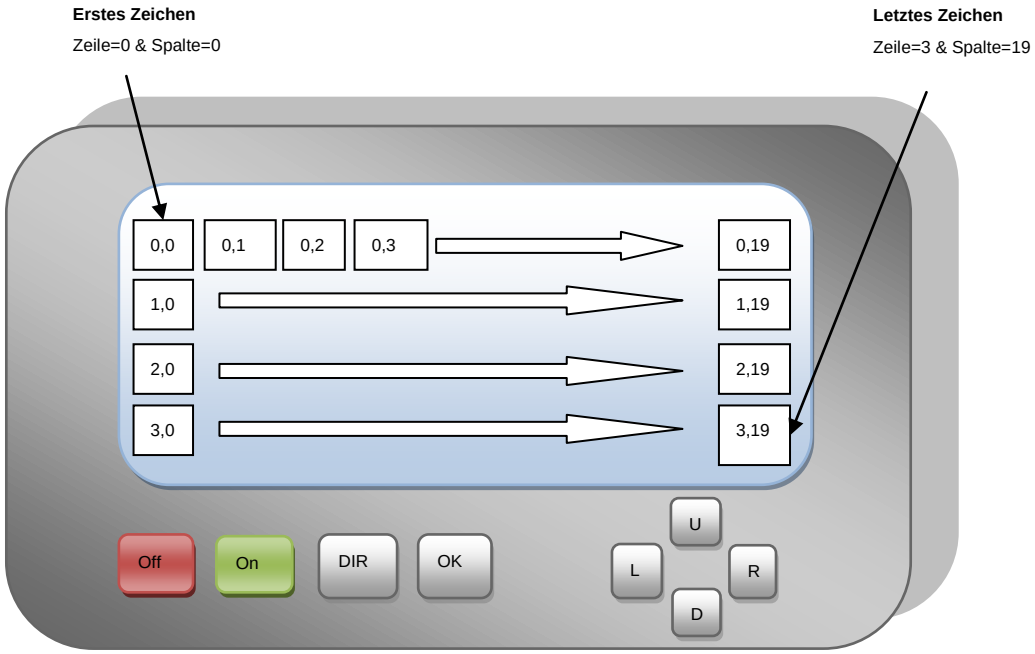



Abbildung 9 Darstellung auf der ParameterBox im Visualisierungsmode

### 3.5.6.1 Überblick Visualisierung

| Funktionsbaustein | Erläuterung |
|---|---|
| FB_STRINGToPBox | Kopiert einen String in die P-Box |
| FB_DINTToPBox | Kopiert einen DINT Wert zur P-Box |

### 3.5.6.2 FB_STRINGToPBox

Dieser Funktionsbaustein kopiert einen String (Zeichenkette) in das P-Box Speicherarray. Über **ROW** und **COLUMN** wird die Startposition des Strings im P-Box Display gesetzt. Der Parameter **TEXT** übergibt den gewünschten String an den Funktionsbaustein, der Stringname kann aus der Variablentabelle entnommen werden.

Solange **ENABLE** auf 1 ist, werden alle Änderungen an den Eingängen sofort übernommen.

Beim gesetzten **CLEAR** Eingang wird der gesamte Display Inhalt mit Leerzeichen überschrieben, bevor der selektierte String geschrieben wird.

Geht **DONE** auf 1, dann ist der String korrekt übertragen worden.

Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig.

Bei einer negativen Flanke an **ENABLE** werden **DONE**, **ERROR** und **ERRORID** zurückgesetzt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Übergabe des Strings | BOOL | **DONE** | String übergeben | BOOL |
| **CLEAR** | Display löschen | BOOL | **ERROR** | Fehler im FB | BOOL |
| **ROW** | Zeile des Displays<br>Wertebereich = 0 bis 3 | BYTE | **ERRORID** | Fehlercode | INT |
| **COLUMN** | Spalte des Displays<br>Wertebereich = 0 bis 19 | BYTE | | | |
| **TEXT** | anzuzeigender Text | STRING | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1500h | String überschreibt den Speicherbereich des P-Box Arrays |
| 1501h | beim Eingang ROW wurde der Wertebereich überschritten |
| 1502h | beim Eingang COLUMN wurde der Wertebereich überschritten |
| 1503h | Die gewählte String Nummer existiert nicht |
| 1506h | In der PLC Konfiguration ist die Option „Parameterbox Funktionsbausteine zulassen“ nicht aktiviert. |

Tabelle 92 FB_STRINGToPBox

### 3.5.6.3 FB_DINTToPBox

Dieser Funktionsbaustein konvertiert einen DINT Wert in einen ASCII String und kopiert diesen in die ParameterBox. Die Ausgabe kann im dezimalen, binären oder hexadezimalen Format erfolgen, die Selektion wird über **MODE** durchgeführt. Über **ROW** und **COLUMN** wird die Startposition des Strings im P-Box Display gesetzt. Der Parameter **LENGTH** übergibt die Länge des Strings in Zeichen. Im **MODE** Dezimal positioniert der Parameter **POINT** ein Komma in die darzustellende Zahl. In **POINT** wird angegeben wie viele Zeichen rechts vom Komma stehen. Bei der Einstellung 0 ist die Funktion **POINT** ausgeschaltet.

Sollte die Zahl mehr Zeichen enthalten als es die Länge zulässt und ist außerdem kein Komma gesetzt, so wird der Überlauf durch das Zeichen „**#**“ angezeigt. Befindet sich ein Komma in der Zahl, so können bei Bedarf alle Zahlen hinter dem Komma entfallen. Im MODE hexadezimal und binär werden immer die niederwertigsten Bits dargestellt, wenn die eingestellte Länge zu kurz ist.

Solange **ENABLE** auf 1 gesetzt ist, werden alle Änderungen an den Eingängen sofort übernommen.

Geht **DONE** auf 1, dann ist der String korrekt übertragen worden.

Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig.

Bei einer negativen Flanke an **ENABLE** werden **DONE**, **ERROR** und **ERRORID** zurückgesetzt.

**Beispiele**:

| Einstellung | Darzustellende Zahl | P-Box Anzeige |
|---|---|---|
| Length = 5<br>Point = 0 | 12345 | 12345 |
| Length = 5<br>Point = 0 | -12345 | ##### |
| Length = 10<br>Point = 3 | 123456789 | 123456,789 |
| Length = 8<br>Point = 3 | 123456789 | 123456,7 |

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Übergabe des Strings | BOOL | **DONE** | String übergeben | BOOL |
| **MODE** | Darstellungsformat<br>0 = Dezimal<br>1 = Binäre<br>2 = Hexadezimal<br>Wertebereich = 0 bis 2 | BYTE | **ERROR** | Fehler im FB | BOOL |
| **ROW** | Zeile des Displays<br>Wertebereich = 0 bis 3 | BYTE | **ERRORID** | Fehlercode | INT |
| **COLUMN** | Spalte des Displays<br>Wertebereich = 0 bis 19 | BYTE | | | |
| **POINT** | Position des Kommas<br>Wertebereich = 0 bis 10<br>0 = Funktion ist ausgeschaltet | BYTE | | | |
| **LENGTH** | Ausgabelänge<br>Wertebereich = 1 bis 11 | BYTE | | | |
| **VALUE** | Auszugebende Zahl | DINT | | | |

| ERRORID | Erläuterung |
|---|---|
| 0 | Kein Fehler |
| 1500h | String überschreibt den Speicherbereich des P-Box Arrays |
| 1501h | beim Eingang LINE wurde der Wertebereich überschritten |
| 1502h | beim Eingang ROW wurde der Wertebereich überschritten |
| 1504h | beim Eingang POINT wurde der Wertebereich überschritten |
| 1505h | beim Eingang LENGTH wurde der Wertebereich überschritten |
| 1506h | beim Eingang MODE wurde der Wertebereich überschritten |

Tabelle 93 FB_DINTToPBox

## 3.5.7 CANopen

Die PLC kann über Funktionsblöcke PDO-Kanäle konfigurieren, überwachen und auf ihnen senden. Über ein PDO können von der PLC bis zu 8 Byte Prozessdaten gesendet oder empfangen werden. Jedes dieser PDO wird über eine eigene Adresse (COB-ID) angesprochen. In der PLC können bis 20 PDO's konfiguriert werden. Zur einfacheren Bedienung wird nicht die COB-ID direkt eingegeben. Stattdessen werden Geräteadresse und die PDO Nummer an den FB übergeben. Die resultierende COB-ID wird auf Basis des Pre-Definded Connection Set (CiA DS301) ermittelt. Dadurch ergeben sich folgende mögliche COB-ID's für die PLC.

| **Sende PDO** | | **Überwachte PDO** | |
|---|---|---|---|
| **PDO** | **COB-ID** | **PDO** | **COB-ID** |
| PDO1 | 200h + Geräteadresse | PDO1 | 180h + Geräteadresse |
| PDO2 | 300h + Geräteadresse | PDO2 | 280h + Geräteadresse |
| PDO3 | 400h + Geräteadresse | PDO3 | 380h + Geräteadresse |
| PDO4 | 500h + Geräteadresse | PDO4 | 480h + Geräteadresse |
| | | | |
| PDO5 | 180h + Geräteadresse | PDO5 | 200h + Geräteadresse |
| PDO6 | 280h + Geräteadresse | PDO6 | 300h + Geräteadresse |
| PDO7 | 380h + Geräteadresse | PDO7 | 400h + Geräteadresse |
| PDO8 | 480h + Geräteadresse | PDO8 | 500h + Geräteadresse |

Tabelle 94 resultierende COB-ID's

NORD Umrichter benutzen zur Prozessdatenübermittlung PDO1, nur für Soll-/Istwert 4 und 5 wird PDO2 verwendet.

### 3.5.7.1 Überblick

| **Funktionsbaustein** | **Erläuterung** |
|---|---|
| FB_PDOConfig | PDO Konfiguration |
| FB_PDOSend | PDO senden |
| FB_PDOReceive | PDO empfangen |
| FB_NMT | PDO freigeben und sperren |

Tabelle 95 Überblick CANopen

### 3.5.7.2 FB_NMT

Nach einem *Power UP* befinden sich alle CAN Teilnehmer im Bus-Zustand Pre-Operational. In diesem Zustand können sie weder PDO empfangen noch senden. Damit die PLC mit anderen Teilnehmern auf dem CAN Bus kommunizieren kann, müssen diese in den Zustand Operational gesetzt werden. Im Regelfall übernimmt dies der Busmaster. Sollte es keinen Busmaster geben, so kann diese Aufgabe vom FB_NMT übernommen werden.

Über die Eingänge **PRE**, **OPE** oder **STOP** kann der Zustand aller am Bus angeschlossenen Teilnehmer beeinflusst werden. Die Eingänge werden mit einer positiven Flanke an **EXECUTE** übernommen. Die Funktion muss solange aufgerufen werden, bis der Ausgang **DONE** oder **ERROR** auf 1 gesetzt wurde.

Wenn der Ausgang **ERROR** auf 1 gesetzt wurde, dann liegt entweder keine 24V Versorgung an der RJ45 CAN Buchse des Umrichters an oder der CAN – Treiber des Umrichters ist im Status *Bus off*.

Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Ausführen | BOOL | **DONE** | NMT Befehl wird gesendet | BOOL |
| **PRE** | Setze alle Teilnehmer in den State Pre-Operational | BOOL | **ERROR** | Fehler im FB | BOOL |
| **OPE** | Setze alle Teilnehmer in den State Operational | BOOL | | | |
| **STOP** | Setze alle Teilnehmer in den State Stopped | BOOL | | | |

Tabelle 96 FB_NMT

### 3.5.7.3 FB_PDOConfig

Über diesen FB werden die PDOs konfiguriert. Mit einer Instanz dieser Funktion können alle gewünschten PDOs konfiguriert werden. Für jedes PDO muss der FB nur einmal aufgerufen werden.

Es können bis zu 20 PDOs eingerichtet werden. Jedes PDO hat seine eigene Parametrierung. Die Zuordnung der PDOs in den anderen CANopen FB's erfolgt über die Messagebox **Number**. Die **TARGETID** stellt die Geräteadresse dar. Bei NORD Frequenzumrichtern wird diese im P515 oder über DIP Schalter eingestellt. Unter **PDO** wird die gewünscht Messagebox-Nummer eingetragen (siehe Einleitung). **LENGTH** legt die Sendelänge eines PDO fest. Über **DIR** wird die Sende-/Empfangsrichtung festgelegt. Mit der positiven Flanke am **EXECUTE** Eingang werden die Daten übernommen.

Der **DONE** Ausgang kann sofort nach Aufruf des FB abgefragt werden. Wenn **DONE** auf 1 gesetzt ist, dann wurde der PDO-Kanal konfiguriert. Bei **ERROR** = 1 gab es ein Problem, die genaue Ursache ist in **ERRORID** abgelegt.

Bei einer negativen Flanke an **EXECUTE** werden alle Ausgänge auf 0 zurückgesetzt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Ausführen | BOOL | **DONE** | PDO konfiguriert | BOOL |
| **NUMBER** | Messagebox Nummer<br>Wertebereich = 0 bis 19 | BYTE | **ERROR** | Fehler im FB | BOOL |
| **TARGETID** | Geräteadresse<br>Wertebereich = 1 bis 127 | BYTE | **ERRORID** | Fehlercode | INT |
| **PDO** | PDO<br>Wertebereich = 1 bis 8 | BYTE | | | |
| **LENGTH** | PDO Länge<br>Wertebereich = 1 bis 8 | BYTE | | | |
| **DIR** | Senden oder Empfangen<br>Senden = 1 / Empfangen = 0 | BOOL | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1800h | Wertebereich Number überschritten |
| 1801h | Wertebereich TARGETID überschritten |
| 1802h | Wertebereich PDO überschritten |
| 1803h | Wertebereich LENGT überschritten |

Tabelle 97 FB_PDOConfig

**HINWEIS**

Es dürfen keine CAN-ID parametriert werden, die der Frequenzumrichter schon benutzt!

Dies betrifft folgende Empfangsadressen:

- CAN ID = 0x180 + P515[01] PDO1
- CAN ID = 0x180 + P515[01] + 1 CAN ID für Absolutwertgeber
- CAN ID = 0x280 + P515[01] PDO2

Dies betrifft folgende Sendeadressen:

- CAN ID = 0x200 + P515[01] PDO1
- CAN ID = 0x300 + P515[01] PDO2

### 3.5.7.4 FB_PDOSend

Mit diesem FB können PDOs auf einem vorher konfigurierten Kanal gesendet werden. Es ist möglich diese einmalig oder zyklisch zu senden. Die zu sendenden Daten werden in **WORD1** bis **WORD4** eingetragen. Ein Senden der PDOs ist unabhängig vom CANopen State des Frequenzumrichters möglich.

Über **NUMBER** wird der vorher konfigurierte PDO Kanal ausgewählt. In **WORD1** bis **WORD4** werden die zu sendenden Daten eingetragen. Über **CYCLE** kann zwischen einmaligem Senden (Einstellung=0) oder zyklischem Senden gewählt werden. Über eine positive Flanke an **EXECUTE** wird das PDO abgeschickt.

Bei **DONE** = 1 waren alle Eingaben korrekt und das PDO wird gesendet. Bei **ERROR** = 1 gab es ein Problem. Die genaue Ursache ist in **ERRORID** abgelegt. Alle Ausgänge werden mit negativer Flanke an **EXECUTE** zurückgesetzt.

Die Zeitbasis der PLC ist 5ms, dies gilt auch für den Eingang **CYCLE**. Es sind nur Sendezyklen mit einem Vielfachen von 5ms realisierbar.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Ausführen | BOOL | **DONE** | PDO gesendet = 1 | BOOL |
| **NUMBER** | Messagebox Nummer<br>Wertebereich = 0 bis 19 | BYTE | **ERROR** | Fehler im FB | BOOL |
| **CYCLE** | Sendezyklus<br>Wertebereich = 0 bis 255<br>0 = ausgeschaltet<br>1 bis 255 = Sendezyklus in ms | BYTE | **ERRORID** | Fehlercode | INT |
| **WORD1** | Sendedaten Wort 1 | INT | | | |
| **WORD2** | Sendedaten Wort 2 | INT | | | |
| **WORD3** | Sendedaten Wort 3 | INT | | | |
| **WORD4** | Sendedaten Wort 4 | INT | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1800h | Wertebereich Number überschritten |
| 1804h | Angewählte Box ist nicht korrekt konfiguriert |
| 1805h | 24V für Bustreiber Fehlen oder Bustreiber ist im State „Bus off“ |

Tabelle 98 FB_PDOSend

Wenn **DONE** auf 1 geht, dann wurde die zu sendende Botschaft vom CAN Modul übernommen, aber noch nicht gesendet. Das eigentliche Senden läuft parallel im Hintergrund. Sollen jetzt über einen FB mehrere Botschaften direkt hintereinander gesendet werden, dann kann es passieren, dass bei dem neuen Aufruf die vorherige Botschaft noch nicht gesendet wurde. Dies kann daran erkannt werden, dass weder das **DONE** noch das **ERROR** Signal nach dem CAL Aufruf auf 1 gesetzt wurde. Der CAL Aufruf kann jetzt einfach so oft wiederholt werden, bis eines der beiden Signale auf 1 geht.

Sollen über einen einzigen FB mehrere verschiedene CAN-ID’s beschrieben werden, so ist dies über eine Neukonfiguration des FB‘s möglich. Diese darf jedoch nicht im selben PLC Zyklus wie das Senden erfolgen, da sonst die Gefahr besteht, dass die zu sendende Botschaft bei der Konfiguration über den FB_PDOConfig gelöscht wird.

### 3.5.7.5 FB_PDOReceive

Dieser FB überwacht einen vorher konfigurierten PDO Kanal auf eingehende Botschaften. Die Überwachung startet wenn der **ENABLE** Eingang auf 1 steht. Nach dem Aufruf der Funktion ist der **NEW** Ausgang zu prüfen. Wenn er auf 1 geht, dann ist eine neue Botschaft angekommen. Der **NEW** Ausgang wird mit dem nächsten Aufruf der Funktion gelöscht.

In **WORD1** bis **WORD4** stehen die empfangenen Daten.

Über **TIME** kann der PDO Kanal auf zyklischen Empfang überwacht werden. Wird in **TIME** ein Wert zwischen 1 und 32767ms eingetragen, dann muss in dieser Zeitspanne eine Botschaft empfangen werden. Anderenfalls geht der FB in den Fehlerzustand (**ERROR** = 1). Über den Wert 0 kann diese Funktion ausgeschaltet werden. Der Überwachungstimer läuft in 5ms Schritten.

Im Fehlerfall wird **ERROR** auf 1 gesetzt. **DONE** ist in diesem Fall 0. In der **ERRORID** ist dann der entsprechende Fehlercode gültig.

Bei einer negativen Flanke an **ENABLE** werden **DONE**, **ERROR** und **ERRORID** zurückgesetzt.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Ausführen | BOOL | **NEW** | Neues PDO empfangen | BOOL |
| **NUMBER** | Messagebox Nummer<br>Wertebereich = 0 bis 19 | BYTE | **ERROR** | Fehler im FB | BOOL |
| **TIME** | Watchdog-Funktion<br>Wertebereich = 0 bis 32767<br>0 = ausgeschaltet<br>1 bis 32767 = Überwachungszeit | INT | **ERRORID** | Fehlercode | INT |
| | | | **WORD1** | Empfangsdaten Wort 1 | INT |
| | | | **WORD2** | Empfangsdaten Wort 2 | INT |
| | | | **WORD3** | Empfangsdaten Wort 3 | INT |
| | | | **WORD4** | Empfangsdaten Wort 4 | INT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1800h | Wertebereich Number überschritten |
| 1804h | Angewählte Box ist nicht korrekt konfiguriert |
| 1805h | 24V für Bustreiber Fehlen oder Bustreiber ist im State „Bus off“ |
| 1807h | Empfangs Timeout ( Watchdog Funktion) |

Tabelle 99 FB_PDOReceipt

**HINWEIS**

Der PLC Zyklus liegt bei 5ms, d.h. bei **einem** Aufruf der Funktion im PLC Programm kann nur alle 5ms eine CAN Botschaft ausgelesen werden. Werden mehrere Botschaften schnell aufeinander gesendet, können Botschaften überschrieben werden.

### 3.5.8 Erfassen schneller Ereignisse ( FB_Capture )

Die Zykluszeit der PLC beträgt 5 ms, dieser Zyklus ist zur Erfassung sehr schneller externer Ereignisse mitunter zu groß. Über der FB Capture ist es möglich auf Flanken an den FU Eingängen bestimmte physikalische Größen zu erfassen. Die Überwachung der Eingänge erfolgt in einem 1 ms Zyklus. Die so gespeicherten Werte können später von der PLC ausgelesen werden.

Bei einer positiven Flanke an **EXECUTE** werden alle Eingänge eingelesen und die Capture Funktion scharf geschaltet. Über den Eingang INPUT wird der zu überwachende FU Eingang selektiert. Über **EDGE** werden die Art der Flanke und das Verhalten des Bausteins ausgewählt.

**EDGE = 0** Mit der ersten positiven Flanke wird der selektierte Wert unter **OUTPUT1** gespeichert und **DONE1** wird auf 1 gesetzt. Die nächste positive Flanke speichert unter **OUTPUT2** und **DONE2** wird auf 1 gesetzt. Der FB wird dann deaktiviert.

**EDGE = 1** Verhalten wie unter **EDGE** = 0, mit dem Unterschied dass die negative Flanke auslöst.

**EDGE = 2** Mit der ersten positiven Flanke wird der selektierte Wert unter **OUTPUT1** gespeichert und **DONE1** wird auf 1 gesetzt. Die nächste negative Flanke speichert unter **OUTPUT2** und **DONE2** wird auf 1 gesetzt. Der FB wird dann deaktiviert.

**EDGE = 3** Verhalten wie unter **EDGE** = 2, mit dem Unterschied das zuerst die negative und dann die positive Flanke auslöst.

Wird der Eingang **CONTINUOUS** auf 1 gesetzt, dann ist für **EDGE** nur noch die Einstellung 0 und 1 relevant. Der FB läuft kontinuierlich weiter und speichert das letzte auslösende Ereignis immer unter **OUTPUT1** ab. **DONE1** bleibt ab dem ersten Ereignis aktiv. DONE2 und OUTPUT2 werden nicht verwendet.

Der **BUSY** Ausgang bleibt solange aktiv bis beide Capture Ereignisse (**DONE1** und **DONE2**) eingetreten sind.

Die Funktion des Bausteins kann jederzeit durch eine negative Flanke an EXECUTE beendet werden. Alle Ausgänge behalten dabei ihre Werte. Mit einer positiven Flanke an **EXECUTE** werden zuerst alle Ausgänge gelöscht.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Ausführen | BOOL | **DONE1** | Wert in OUTPUT1 gültig | BOOL |
| **CONTINUOUS** | Einmalige Ausführung o. Dauerbetrieb | BOOL | **DONE2** | Wert in OUTPUT2 gültig | BOOL |
| **INPUT** | Zu überwachender Eingang<br>0 = Eingang 1<br>- - - -<br>7 = Eingang 8 | BYTE | **BUSY** | FB wartet noch auf Capture Ereignisse | BOOL |
| **EDGE** | Auslösende Flanke | BYTE | **ERROR** | der FB hat einen Fehler | BOOL |
| **SOURCE** | Zu speicherende Größe<br>0 = Position in Umdrehungen<br>1 = Istfrequenz<br>2 = Moment | BYTE | **ERRORID** | Fehlercode | INT |
| | | | **OUTPUT1** | Wert für 1. Capture Ereignisses | DINT |
| | | | **OUTPUT2** | Wert für 2. Capture Ereignisses | DINT |

| ERRORID | Erläuterung |
|---|---|
| 0 | Kein Fehler |
| 1900h | Wertebereich INPUT wurde überschritten |
| 1901h | Wertebereich EDGE wurde überschritten |
| 1902h | Wertebereich SOURCE wurde überschritten |
| 1903h | Es sind mehr als zwei Instanzen aktiv |

Tabelle 100 FB_Capture

**HINWEIS**

Von diesem FB können mehrere Instancen im PLC Programm existieren. Aber es dürfen zur selben Zeit nur zwei Instancen aktiv sein!

## 3.5.9 Zugriff auf Speicherbereiche des Frequenzumrichters

Wenn es nötig ist, größere Mengen an Daten zwischen zu speichern, an andere Geräte zu übergeben oder von anderen Geräten zu empfangen, dann ist die Verwendung der Bausteine FB_WriteTrace und FB_ReadTrace angebracht.

### 3.5.9.1 Überblick Speicherbausteine

| Funktionsbaustein | Erläuterung |
|---|---|
| FB_WriteTrace | Speichert einzelne oder auch größere Datenmengen |
| FB_ReadTrace | Liest einzelne oder auch größere Datenmengen |

### 3.5.9.2 FB_WriteTrace

Über diesen FB können einzelne oder auch größere Mengen an Werten im FU zwischengespeichert werden. Das Speichern der Werte erfolgt nicht dauerhaft, d.h. nach einem Neustart des FU gehen die Werte verloren.

Wird vom FB eine positive Flanke am **ENABLE** Eingang erkannt, dann werden alle am Eingang anliegenden Parameter übernommen. Der in **VALUE** stehende Wert wird auf die durch **STARTINDEX** und **MEMORY** gekennzeichnete Speicherstelle geschrieben. Bei einem erfolgreichen Schreibvorgang geht der Ausgang **VALID** auf 1.

Wird der FB jetzt mehrfach aufgerufen und der **ENABLE** Eingang bleibt auf 1, dann wird bei jedem FB Aufruf der Eingang **VALUE** gelesen und gespeichert, sowie die Speicheradresse um 1 erhöht. Der aktuelle Speicherindex für den nächsten Zugriff kann unter dem Ausgang **ACTINDEX** ausgelesen werden. Wird das Speicherende erreicht, dann geht der Ausgang **FULL** auf 1 und der Speichervorgang wird gestoppt. Ist jedoch der Eingang **OVERWRITE** auf 1 gesetzt ist, so wird der Speicherindex wieder auf den **STARTINDEX** gesetzt und es werden die vorher gespeicherten Werte überschrieben.

Es können Werte im INT oder DINT Format gespeichert werden. Bei INT Werten, wird vom Eingang **VALUE** nur der Low Teil ausgewertet. Die Zuordnung erfolgt über den Eingang **SIZE**, eine 0 steht für INT und eine 1 für DINT Werte.

Die Zuordnung der Speicherbereiche erfolgt über den Eingang **MEMORY**:
**MEMORY** = 1 → P613[0-251] entspricht 504 INT oder 252 DINT Werten
**MEMORY** = 0 → P900[0-247] bis P906[0-111] entspricht 5200 INT oder 2600 DINT Werten

Der FB kann nicht durch andere Blöcke unterbrochen werden.

Mit einer negativen Flanke an **ENABLE** werden alle Ausgänge auf 0 gesetzt und die Funktion des FB beendet.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Ausführen | BOOL | **VALID** | Schreibvorgang erfolgreich | BOOL |
| **SIZE** | Speicherformat | BOOL | **FULL** | Komplette Speicher ist voll | BOOL |
| **OVERWRITE** | Speicher überschreibar | BOOL | **ERROR** | der FB hat einen Fehler | BOOL |
| **MEMORY** | Auswahl Speicherbereich | BYTE | **ERRORID** | Fehlercode | INT |
| **STARTINDEX** | Zeigt auf die zu beschreibende Speicherzelle | INT | **ACTINDEX** | Aktueller Speicherindex, auf dem im nächsten Zyklus gespeichert wird | DINT |
| **VALUE** | Zu speichernder Wert | DINT | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1A00h | Wertebereich STARTINDEX wurde überschritten |
| 1A01h | Wertebereich MEMORY wurde überschritten |

Tabelle 101 FB_WriteTrace

**HINWEIS**

Achtung! Der Speicherbereich in der Einstellung **MEMORY** = 0 wird auch von der Scope Funktion genutzt. Ein Verwenden der Scope Funktion überschreibt die gespeicherten Werte!

### 3.5.9.3 FB_ReadTrace

Mit Hilfe dieses FB können verschiedene Speicherbereiche des FU direkt ausgelesen werden.

Wird vom FB eine positive Flanke am **ENABLE** Eingang erkannt, dann werden alle am Eingang anliegenden Parameter übernommen. Durch **STARTINDEX** und **MEMORY** wird die auszulesende Speicherstelle gekennzeichnet. Bei einem erfolgreichen Lesevorgang geht der Ausgang **VALID** auf 1 und in **VALUE** steht der ausgelesene Wert.

Wird der FB jetzt mehrfach aufgerufen und der **ENABLE** Eingang bleibt auf 1, dann wird bei jedem Aufruf die auszulesende Speicheradresse um 1 erhöht, der Inhalt der neue Speicherzelle sofort in den Ausgang **VALUE** kopiert.

Der aktuelle Speicherindex für den nächsten Zugriff kann unter dem Ausgang **ACTINDEX** ausgelesen werden. Wird das Speicherende erreicht, dann geht der Ausgang **READY** auf 1 und der Lesevorgang wird gestoppt.

Es können Werte im INT oder DINT Format gelesen werden. Bei INT Werten, ist vom Ausgang **VALUE** nur der Low Teil auszuwerten. Die Zuordnung erfolgt über den Eingang **SIZE**, eine 0 steht für INT und eine 1 für DINT Werte.

Die Zuordnung der Speicherbereiche erfolgt über den Eingang **MEMORY**:
**MEMORY** = 1 → P613[0-251] entspricht 504 INT oder 252 DINT Werten
**MEMORY** = 0 → P900[0-247] bis P906[0-111] entspricht 3200 INT oder 1600 DINT Werten

Der FB kann nicht durch andere Blöcke unterbrochen werden.

Mit einer negativen Flanke an **ENABLE** werden alle Ausgänge auf 0 gesetzt und die Funktion des FB beendet.

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **ENABLE** | Ausführen | BOOL | **VALID** | Lesevorgang erfolgreich | BOOL |
| **SIZE** | Speicherformat | BOOL | **READY** | Der gesamte Speicher ist ausgelesen | BOOL |
| **MEMORY** | Auswahl Speicherbereich | BYTE | **ERROR** | der FB hat einen Fehler | BOOL |
| **STARTINDEX** | Zeigt auf die zu beschreibende Speicherzelle | INT | **ERRORID** | Fehlercode | INT |
| | | | **ACTINDEX** | Aktueller Speicherindex, aus dem im nächsten Zyklus gelesen wird | INT |
| | | | **VALUE** | Ausgelesener Wert | DINT |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 1A00h | Wertebereich STARTINDEX wurde überschritten |
| 1A01h | Wertebereich MEMORY wurde überschritten |

Tabelle 102 FB_ReadTrace

### 3.5.10 Wiegefunktion (FB_Weigh)

Dieser Baustein dient zur Ermittlung des durchschnittlichen Drehmoments während einer Fahrt mit konstanter Drehzahl. Aus diesem Wert können dann z.B. physikalische Größen wie das bewegte Gewicht ermittelt werden.

Über eine positive Flanke am **EXECUTE** Eingang wird der FB gestartet. Mit der Flanke werden alle Eingänge am FB übernommen. Der FU verfährt mit der unter **SPEED** gesetzten Drehzahl. Nach Ablauf der unter **STARTTIME** gesetzten Zeit wird mit der Messung begonnen. Die Messdauer wird unter **MEASURETIME** definiert. Nach Ablauf der Messzeit stoppt der FU. Wenn der Eingang **REVERSE** = 1 ist, dann startet der Messvorgang erneut jedoch mit negierter Drehzahl. Ansonsten ist die Messung beendet, der Ausgang **DONE** geht auf 1 und in **VALUE** steht das Messergebnis.

Solange der Messvorgang läuft ist **BUSY** aktiv.

Die Skalierung des Messergebnis **VALUE** ist 1 = 0,01% vom Nenndrehmoment des Motors.

Der Aufruf eines anderen Motion FB stoppt die Messfunktion und der Ausgang **ABORT** geht auf 1.

Alle Ausgänge des FB werden mit einer neuen positiven Flanke an **EXECUTE** rückgesetzt (Reset).

| VAR_INPUT | | | VAR_OUTPUT | | |
|---|---|---|---|---|---|
| **Eingang** | **Erläuterung** | **Typ** | **Ausgang** | **Erläuterung** | **Typ** |
| **EXECUTE** | Ausführen | BOOL | **DONE** | Messung beendet | BOOL |
| **REVERSE** | Drehrichtungswechsel | BOOL | **BUSY** | Messung läuft | BOOL |
| **STARTTIME** | Zeit bis Messbegin in ms (<2000ms) | INT | **ABORT** | Messung abgebrochen | BOOL |
| **MEASURETIME** | Messzeit in ms (60 bis 2000ms) | INT | **ERROR** | der FB hat einen Fehler | BOOL |
| **SPEED** | Messgeschwindigkeit in % (16#4000 entspricht 100%) | INT | **ERRORID** | Fehlercode | INT |
| | | | **VALUE** | Meßergebnis | INT |
| | | | | | |

| **ERRORID** | **Erläuterung** |
|---|---|
| 0 | Kein Fehler |
| 0x1000 | FU nicht eingeschaltet |
| 0x1101 | Sollfrequenz nicht als Sollwert parametriert (P553) |
| 0x1C00 | Wertebereich STARTTIME wurde überschritten |
| 0x1C01 | Wertebereich MEASURETIME wurde überschritten |
| 0x1C02 | Die Toleranz der Messwerte zueinander, ist größer als 1/8 |

Tabelle 103 FB_Weigh

**HINWEIS**

Von diesem FB ist nur eine Instance im PLC Programm zulässig!

# 4 Details Frequenzumrichter

## 4.1 Parameter

In diesem Kapitel werden nur Parameter und Einstellungen aufgeführt, die im direkten Zusammenhang mit der PLC Funktion stehen. Die vollständige Parameterliste ist dem Handbuch des Frequenzumrichters BU0500 / BU0505 zu entnehmen.

### 4.1.1 Betriebsanzeigen

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P001** | **Auswahl Anzeige**<br>*(Auswahl Anzeige)* | | | |
| 0 ...65<br>{ 0 } | **Einstellung:**<br>**…**<br>**40 = PLC-Ctrlbox Wert,** Visualisierungsmodus für PLC-Kommunikation<br>**…** | | | |

### 4.1.2 Reglungs- / PLC Parameter

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P350** | **PLC – Funktionalität**<br>*(PLC – Funktionalität)* | | | P |
| 0 … 1<br>{ 0 } | **Einstellung:**<br>**0 = Aus**, die PLC ist nicht aktiv, die Ansteuerung des Frequenzumrichters erfolgt gemäß Parameter (P509) und (P510)<br>**1 = An**, die PLC ist aktiv, die Ansteuerung des Frequenzumrichters erfolgt in Abhängigkeit von (P351) über die PLC. Die Definition der Hauptsollwerte ist dementsprechend im Parameter (P553) vorzunehmen. Nebensollwerte (P510[-02]) können weiterhin über (P546) definiert werrden. | | | |
| **P351** | **PLC Sollwert Auswahl**<br>*(PLC Sollwert Auswahl)* | | | P |
| 0 … 3<br>{ 0 } | Auswahl der Quelle für STW und HSW bei aktiver PLC – Funktionalität (P350 = 1). Bei Einstellung „0“ und „1“ erfolgt die Definition der Hauptsollwerte über (P553), die der Nebensollwerte jedoch unverändert über (P546). Dieser Parameter wird nur übernommen, wenn der Frequenzumrichter sich im State „Einschaltbereit“ befindet.<br>**Einstellung:**<br>**0 = STW und HSW =PLC,** Die PLC liefert Steuerwort und Hauptsollwert, die Parameter (P509) und (P510[-01]) haben keine Funktion.<br>**1 = STW=P509,** Die PLC liefert den Hauptsollwert, die Steuerwortquelle entspricht der Einstellung in Parameter (P509)<br>**2 = HSW=P510[-01],** Die PLC liefert das Steuerwort, die Quelle für den Hauptsollwert entspricht der Einstellung in Parameter (P510[-01])<br>**3 = STW und HSW=P509/P510 ,** Die Quelle für Steuerwort und Hauptsollwert entspricht der Einstellung in Parameter (P509)/(P510[-01]) | | | |

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P353** | **Buszustand über PLC**<br>*(Buszustand über PLC)* | | | |
| 0 … 3<br>{ 0 } | Über diesen Parameter kann entschieden werden, wie das Steuerwort für die Leitfunktion und das Zustandswort des Frequenzumrichters von der PLC weiterverarbeitet werden.<br>**Einstellung:**<br>**0 =** **Aus,** Steuerwort der Leitfunktion (P503≠0) und Zustandswort werden von der PLC unverändert weiterverarbeitet.<br>**1 =** **STW für Leitfunktion durch PLC**, das Steuerwort für die Leitwertfunktion (P503≠ 0) wird durch die PLC gesetzt. Dazu ist in der PLC mittels Prozesswert „34_PLC_Busmaster_Control_word“ (Kap. 4.2.2 ) das Steuerword entsprechend neu zu definieren.<br>**2 =** **ZSW für BUS durch PLC**, das Zustandswort des Frequenzumrichters wird durch die PLC gesetzt. Dazu ist in der PLC mittels Prozesswert „28_PLC_status_word“ (Kap. 4.2.2 ) das Zustandswort entsprechend neu zu definieren.<br>**3 =** **STW und ZSW durch PLC,** *siehe Einstellung 1 und 2* | | | |
| **P355** **… [-01]**<br>**…**<br>**… [-10]** | **PLC – Integer Sollwert**<br>*(PLC – Integer Sollwert)* | | | |
| 0x0000 … 0xFFFF<br>alle { 0 } | Über dieses Int Array können mit der PLC Daten ausgetauscht werden. Diese Daten können durch die entsprechenden Prozessvariablen in der PLC verwendet werden. | | | |
| **P356** **… [-01]**<br>**…**<br>**… [-05]** | **PLC - DINT – Sollwert**<br>*(PLC - DINT – Sollwert)* | | | |
| 0x0 … 0xFFFFFFFF<br>alle { 0 } | Über dieses DINT Array können mit der PLC Daten ausgetauscht werden. Diese Daten können durch die entsprechenden Prozessvariablen in der PLC verwendet werden. | | | |
| **P360** **… [-01]**<br>**…**<br>**… [-05]** | **PLC – Anzeigewert**<br>*(PLC – Anzeigewert)* | | | |
| -2.000,000 …<br>2.000,000<br>alle { 0 } | Dient nur zur Anzeige von PLC Daten mit einer Auflösung von 0,001. Durch die entsprechenden Prozessvariablen können diese Parameter von der PLC beschrieben werden. Die Werte werden nicht gespeichert! | | | |
| **P370** | **PLC – Status**<br>*(PLC – Status)* | | | |
| 0 … $63_{dez}$<br>**ParameterBox:**<br>0 … $3F_{hex}$<br>**Simple-/ ControlBox:**<br>0 …. 11 $1111_{bin}$ | Zustand der PLC<br>**Bit 0 = P350 = 1**<br>**Bit 1 = PLC aktiv**<br>**Bit 2 = Stop aktiv**<br>**Bit 3 = Debug aktiv**<br>**Bit 4 =** **PLC Fehler,** die PLC User Fehler 23.xx werden hier nicht angezeigt<br>**Bit 5 =** **PLC angehalten,** Single Step oder Breakpoint | | | |

### 4.1.3 Steuerklemmen

| Parameter<br>{ Werkseinstellung } | | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|---|
| **P418** | **... [-01]**<br>**...**<br>**... [-03]** | **Funkt. Analogausg.**<br>*(Funktion Analogausgang)* | | S | P |

0 ... 60

alle { 0 }

**analoge Funktionen** (max. Last: 5mA analog)**:**

An den Steuerklemmen kann eine analoge (0 ... +10 Volt) Spannung abgenommen werden (max. 5mA). Für die PLC Einstellung gilt folgende Skalierung:

- PLC Wert : 0 = 0V
- PLC Wert : 100 = 10.0V

**[-01] = Analogausgang** (int. AOUT), **keine Funktion** als Werkseinstellung, AOUT des Frequenzumrichters, Steuerklemme 17

**[-02] = Erste IOE** (ext. AOUT1), **keine Funktion** als Werkseinstellung, AOUT der 1. IO-Erweiterungsbaugruppe (SK xU4-IOE)

**[-03] = Zweite IOE** (ext. AOUT2), **keine Funktion** als Werkseinstellung, AOUT der 2. IO-Erweiterungsbaugruppe (SK xU4-IOE)

**Einstellung:**

**...**

**60 = Wert von PLC,** der Wert kommt von der PLC

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P420 … [-01]**<br>**…**<br>**… [-10]** | **Digitaleingänge**<br>*(Funktion Digitaleingänge)* | | | |

0 … 80

[-01] = { 1 }
[-02] = { 2 }
[-03] = { 8 }
[-04] = { 4 }
alle anderen = { 0 }

Im SK 54xE stehen bis zu 10 frei programmierbare digitale Eingänge zur Verfügung. Mit der unten aufgeführten Einstellung kann die PLC Programmausführung gestoppt werden.

**[-01] =** **Digitaleingang 1** (DIN1), **Freigabe rechts** als Werkseinstellung, Steuerklemme 21

**[-02] =** **Digitaleingang 2** (DIN2), **Freigabe links** als Werkseinstellung, Steuerklemme 22

**[-03] =** **Digitaleingang 3** (DIN3), **Parametersatzumschaltung Bit 0** als Werkseinstellung, Steuerklemme 23

**[-04] =** **Digitaleingang 4** (DIN4), **Festfrequenz 1** als Werkseinstellung, Steuerklemme 24

**[-05] =** **Digitaleingang 5** (DIN5), **keine Funktion** als Werkseinstellung, Steuerklemme 25 (nur bei Geräten ab BG5 vorhanden)

**[-06] =** **Digitaleingang 6** (DIN6), **keine Funktion** als Werkseinstellung, Steuerklemme 26

**[-07] =** **Digitaleingang 7** (DIN7), **keine Funktion** als Werkseinstellung, Steuerklemme 27, (DIN7 kann auch als DOUT3 verwendet werden)

**[-08] =** **Digitalfunkt. Analog1** (AIN1), **keine Funktion** als Werkseinstellung, Steuerklemme 14, (Digitalfunktion Analogeingang AIN1)

**[-09] =** **Digitalfunkt. Analog2** (AIN2), **keine Funktion** als Werkseinstellung, Steuerklemme 16, (Digitalfunktion Analogeingang AIN2)

**[-10] =** **Digitaleingang 8** (DIN8), **keine Funktion** als Werkseinstellung, Steuerklemme 7, (DIN8 kann auch als DOUT2 verwendet werden)

**Einstellung:**

**…**

**80 =** **PLC-Stop,** die PLC – Programmausführung wird gestoppt, solange das Signal anliegt

| | | | | |
|---|---|---|---|---|
| **P434 … [-01]**<br>**…**<br>**… [-05]** | **Digitalausgang Funkt.**<br>*(Funktion Digitalausgänge)* | | | P |

0 … 40

[-01] = { 1 }
[-02] = { 7 }
alle anderen = { 0 }

Die digitalen Ausgänge können bei der Einstellung PLC über die selbige gesetzt werden.

**[-01] =** **Binärausg. 1 / (MFR1)**, **externe Bremse** als Werkseinstellung, Steuerklemme 1/2

**[-02] =** **Binärausg. 2 / (MFR2)**, **Störung** als Werkseinstellung, Steuerklemme 3/4

**[-03] =** **Binärausg. 3 / (DOUT1)**, **keine Funktion** als Werkseinstellung, Steuerklemme 5

**[-04] =** **Binärausg. 4 / (DOUT2)**, **keine Funktion** als Werkseinstellung, Steuerklemme 7

**[-05] =** **Binärausg. 5 / (DOUT3)**, **keine Funktion** als Werkseinstellung, Steuerklemme 27, (DOUT3 kann auch als DIN7 verwendet werden)

**Einstellung:**

**…**

**40 =** **Ausgang über PLC,** Ausgang wird durch die PLC gesetzt

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P480 … [-01]<br>…<br>… [-12]** | **Funkt. BusIO In Bits**<br>*(Funktion Bus I/O In Bits)* | | S | |
| 0 … 80<br>alle { 0 } | Die Bus I/O In Bits werden wie Digitaleingänge angesehen. Sie können auf die gleichen Funktionen (P420) eingestellt werden.<br>Diese I/O Bits können auch im Zusammenhang mit den I/O-Erweiterungen (SK xU4-IOE) (Bit 4 … 7 und Bit 0 … 3) genutzt werden.<br>**[-01] = BusIn Bit0/IOE2 DIN1,** Bus IO In Bit 0 bzw. DI 1 der **zweiten** SK xU4-IOE<br>**[-02] = BusIn Bit1/IOE2 DIN2,** Bus IO In Bit 1 bzw. DI 2 der **zweiten** SK xU4-IOE<br>**[-03] = BusIn Bit2/IOE2 DIN3,** Bus IO In Bit 2 bzw. DI 3 der **zweiten** SK xU4-IOE<br>**[-04] = BusIn Bit3/IOE2 DIN4,** Bus IO In Bit 3 bzw. DI 4 der **zweiten** SK xU4-IOE<br>**[-05] = BusIn Bit4/IOE1 DIN1,** Bus IO In Bit 4 bzw. DI 1 der **ersten** SK xU4-IOE<br>**[-06] = BusIn Bit5/IOE1 DIN2,** Bus IO In Bit 5 bzw. DI 2 der **ersten** SK xU4-IOE<br>**[-07] = BusIn Bit6/IOE1 DIN3,** Bus IO In Bit 6 bzw. DI 3 der **ersten** SK xU4-IOE<br>**[-08] = BusIn Bit7/IOE1 DIN4,** Bus IO In Bit 7 bzw. DI 4 der **ersten** SK xU4-IOE<br>**[-09] = Merker 1**<br>**[-10] = Merker 2**<br>**[-11] = Bit 8 BUS Steuerwort**<br>**[-12] = Bit 9 BUS Steuerwort**<br><br>**Einstellung:**<br>**…**<br>**80 = PLC-Stop,** die PLC – Programmausführung wird gestoppt, solange das Signal anliegt | | | |
| **P481 … [-01]<br>…<br>… [-10]** | **Funkt. BusIO Out Bits**<br>*(Funktion Bus I/O Out Bits)* | | S | |
| 0 … 40<br>alle { 0 } | Die Bus I/O In Bits werden wie Digitalausgänge angesehen. Sie können auf die gleichen Funktionen (P434) eingestellt werden.<br>Diese I/O Bits können auch im Zusammenhang mit den I/O-Erweiterungen (SK xU4-IOE) (Bit 4 … 5 und Merker 1 … 2) genutzt werden.<br>**[-01] = BusOut Bit0,** Bus IO Out Bit 0<br>**[-02] = BusOut Bit1,** Bus IO Out Bit 1<br>**[-03] = BusOut Bit2,** Bus IO Out Bit 2<br>**[-04] = BusOut Bit3,** Bus IO Out Bit 3<br>**[-05] = BusOut Bit4/IOE1 DO1,** Bus IO Out Bit 4 + DO1 der **ersten** SK xU4-IOE<br>**[-06] = BusOut Bit5/IOE1 DO2,** Bus IO Out Bit 5 + DO2 der **ersten** SK xU4-IOE<br>**[-07] = BusOut Bit6/IOE2 DO1,** Bus IO Out Bit 6 + DO1 der **zweiten** SK xU4-IOE<br>**[-08] = BusOut Bit7/IOE2 DO2,** Bus IO Out Bit 7 + DO2 der **zweiten** SK xU4-IOE<br>**[-09] = Bit10 BUS Statuswort**<br>**[-10] = Bit13 BUS Statuswort**<br><br>**Einstellung:**<br>**…**<br>**40 = Ausgang über PLC,** Bit wird durch die PLC gesetzt | | | |

### 4.1.4 Zusatzparameter

| Parameter { Werkseinstellung } | | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|---|
| **P502** | **… [-01]**<br>**…**<br>**… [-05]** | **Wert Leitfunktion**<br>*(Wert Leitfunktion)* | | S | P |
| 0 … 58<br>alle { 0 } | | Auswahl der bis zu 5 Leitwerte:<br>**[-01] = Leitwert 1** **…** **[-05] = Leitwert 5**<br><br>**Einstellung:**<br>**…**<br>**53 = PLC_status_word** **56 = PLC_act_val3**<br>**54 = PLC_act_val1** **57 = PLC_act_val4**<br>**55 = PLC_act_val2** **58 = PLC_act_val5** | | | |
| **P503** | | **Leitfunktion Ausgabe**<br>*(Leitfunktion Ausgabe)* | | S | |
| 0 ... 5<br>{ 0 } | | Zur Nutzung der Leitfunktion Ausgabe ist im P509 die Quelle der Umrichtersteuerung zu wählen. Im Parameter P502 wird der über das BUS-Interface zu übertragende Leitwert bestimmt.<br>**0 = Aus** **1 = USS** **2 = CAN** (bis zu 250kBaud) **3 = CANopen**<br>**4 = Systembus aktiv** **5 = CANopen+Sys.bus akt.** | | | |
| **P509** | | **Quelle Steuerwort**<br>*(Quelle Steuerwort)* | | | |
| 0 ... 10<br>{ 0 } | | Auswahl der Schnittstelle, über die der FU angesteuert wird (Details siehe BU0500). Beachte auch Parameter (P350).<br>**0 = Steuerklemmen oder Tastatur** **6 = CANopen**<br>**1 = Steuerklemmen** **7 = DeviceNet**<br>**2 = USS (**bzw. **Modbus RTU** ab SK 540E**)** **8 = Ethernet TU**<br>**3 = CAN** **9 = CAN Broadcast**<br>**4 = Profibus** **10 = CANopen Broadcast**<br>**5 = InterBus** | | | |
| **P510** | **[-01]**<br>**[-02]** | **Quelle Sollwerte**<br>*(Quelle Sollwerte)* | | S | |
| 0 ... 10<br>{ alle 0 } | | Auswahl der zu parametrierenden Sollwertquelle (Details siehe BU0500). Beachte auch Parameter (P350) und (P351).<br>**[-01] =** Quelle **Hauptsollwert** **[-02] =** Quelle **Nebensollwert**<br><br>**0 = Auto** **6 = CANopen**<br>**1 = Steuerklemmen** **7 = DeviceNet**<br>**2 = USS (**bzw. **Modbus RTU** ab SK 540E**)** **8 = Ethernet TU**<br>**3 = CAN** **9 = CAN Broadcast**<br>**4 = Profibus** **10 = CANopen Broadcast**<br>**5 = InterBus** | | | |

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P543** **… [-01]**<br>**…**<br>**… [-05]** | **Bus – Istwert**<br>*(Bus – Istwerte)* | | S | P |
| 0 … 57<br><br>[-01] = { 1 }<br>[-02] = { 4 }<br>[-03] = { 9 }<br>alle anderen = { 0 } | In diesem Parameter kann der Rückgabewert bei Busansteuerung gewählt werden.<br>**[-01] = Bus - Istwert 1 … [-05] = Bus - Istwert 5**<br><br>**Einstellung:**<br>**…**<br>**53 = Istwert 1 PLC**<br>**54 = Istwert 2 PLC**<br>**55 = Istwert 3 PLC**<br>**56 = Istwert 4 PLC**<br>**57 = Istwert 5 PLC** | | | |
| **P553** **… [-01]**<br>**…**<br>**… [-05]** | **PLC Sollwerte**<br>*(PLC Sollwerte)* | | | |
| 0 … 57<br><br>alle = { 0 } | In diesem Parameter wird den PLC Sollwerten eine Funktion zugeordnet. Die Einstellungen gelten nur für Hauptsollwerte und bei aktiver PLC - Ansteuerung ((P350) = „An“) und ((P351)= „0“ oder „1“).<br>**[-01] = PLC Sollwert 1 … [-05] = PLC Sollwert 5**<br><br>**Einstellung:**<br>**0 = Aus**<br>**1 = Sollfrequenz**<br>**2 = Momentstromgrenze**<br>**3 = Istfrequenz PID**<br>**4 = Frequenzaddition**<br>**5 = Frequenzsubtrakt.**<br>**6 = Stromgrenze**<br>**7 = Maximalfrequenz**<br>**8 = IstFreq PID begrenzt**<br>**9 = IstFreq PID überwacht**<br>**10 = Drehmoment Servomode**<br>**11 = Vorhalt Drehmoment**<br>**12 =** reserviert<br>**13 = Multiplikation**<br>**14 = Istwert Prozeßregler**<br>**15 = Sollwert Prozeßregl.**<br>**16 = Vorhalt Prozeßregler**<br>**17 = BusIO In Bits 0-7**<br>**18 = Kurvenfahrtrechner**<br>**19 = Relais setzen**<br>**20 = Analogausg. setzen**<br>**21 = Sollposition LowWord**<br>**22 = Sollpos. HighWord**<br>**23 = Sollpos. Ink.LowWord**<br>**24 = Sollpos. Ink.HighWord**<br>**25 … 45 =** reserviert<br>**46 = Sollw.Drehm.Pzregl.**<br>**47 = Über.-faktor Gearing**<br>**48 = Motortemperatur**<br>**49 = Rampenzeit**<br>**50 … 52 =** reserviert<br>**53 = d-Korr. F Prozess**<br>**54 = d-Korr. Drehmoment**<br>**55 = d-Korr. F+Drehm.**<br>**56 = Beschleunigungszeit**<br>**57 = Bremszeit**<br>Details zu den einzelnen Einstellungen: siehe Handbuch BU0500 (z.B. im Parameter P400) | | | |

### 4.1.5 Informationsparameter

| Parameter { Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P740** **… [-01]** … **… [-23]** | **Prozeßdaten Bus In** *(Prozessdaten Bus In)* | | S | |
| 0 ... FFFF (hex) | Dieser Parameter informiert über das aktuelle Steuerwort und die Sollwerte, die über das Bussystem oder die PLC eingebracht werden. | | | |
| | **[-01] = Steuerwort (P509)** | Steuerwort, Quelle aus (P509). | | |
| | **[-02] = Sollwert 1 (P510 [-01])** / (P546 [-01])<br>**[-03] = Sollwert 2 (P510 [-01])** / (P546 [-02])<br>**[-04] = Sollwert 3 (P510 [-01])** / (P546 [-03])<br>**[-05] = Sollwert 4 (P510 [-01])** / (P546 [-04])<br>**[-06] = Sollwert 5 (P510 [-01])** / (P546 [-05]) | Sollwertdaten vom Hauptsollwert (P510 [-01]). | | |
| | **[-07] = res.Zust.InBit P480** | „resultierender Zustand In Bit P480“, Der angezeigte Wert stellt alle Bus In Bit Quellen mit *oder* verknüpft dar. | | |
| | **[-08] = Parameterdaten In 1**<br>**[-09] = Parameterdaten In 2**<br>**[-10] = Parameterdaten In 3**<br>**[-11] = Parameterdaten In 4**<br>**[-12] = Parameterdaten In 5** | Daten bei Parameterübertragung: Auftragskennung (AK), Parameternummer (PNU), Index (IND), Parameterwert (PWE1/2). | | |
| | **[-13] = Sollwert 1 (P510 [-02])**<br>**[-14] = Sollwert 2 (P510 [-02])**<br>**[-15] = Sollwert 3 (P510 [-02])**<br>**[-16] = Sollwert 4 (P510 [-02])**<br>**[-17] = Sollwert 5 (P510 [-02])** | Sollwertdaten vom Leitfunktions-Wert (Broadcast), wenn P509/510 = 9 oder 10 (P502/P503). | | |
| | **[-18] = Steuerwort PLC** | Steuerwort, Quelle PLC. | | |
| | **[-19] = Sollwert 1 PLC**<br>**[-20] = Sollwert 2 PLC**<br>**[-21] = Sollwert 3 PLC**<br>**[-22] = Sollwert 4 PLC**<br>**[-23] = Sollwert 5 PLC** | Sollwertdaten von der PLC. | | |

| Parameter<br>{ Werkseinstellung } | Einstellwert / Beschreibung / Hinweis | Gerät | Supervisor | Parameter-satz |
|---|---|---|---|---|
| **P741 ... [-01]**<br>**...**<br>**... [-23]** | **Prozeßdaten Bus Out**<br>*(Prozessdaten Bus Out)* | | S | |

0 ... FFFF (hex)

Dieser Parameter informiert über das aktuelle Statuswort und die Istwerte, die über das Bussystem übertragen werden.

| | | |
|---|---|---|
| **[-01] =** | **Státuswort Bus (P509)** | Statuswort |
| **[-02] =** | **Bus-Istwert 1 (P543 [-01])** | Istwerte |
| **[-03] =** | **Bus-Istwert 2 (P543 [-02])** | |
| **[-04] =** | **Bus-Istwert 3 (P543 [-03])** | |
| **[-05] =** | **Bus-Istwert 4 (P543 [-04])** | |
| **[-06] =** | **Bus-Istwert 5 (P543 [-05])** | |
| **[-07] =** | **res.Zust.OutBit P481** | „resultierender Zustand Out Bit P481“, Der angezeigte Wert stellt alle Bus Out Bit Quellen mit *oder* verknüpft dar. |
| **[-08] =** | **Parameterdaten Out 1** | Daten bei Parameterübertragung: Auftragskennung (AK), Parameternummer (PNU), Index (IND), Parameterwert (PWE1/2). |
| **[-09] =** | **Parameterdaten Out 2** | |
| **[-10] =** | **Parameterdaten Out 3** | |
| **[-11] =** | **Parameterdaten Out 4** | |
| **[-12] =** | **Parameterdaten Out 5** | |
| **[-13] =** | **Istwert 1 Leitfunktion** (P502[-01]) | Istwertdaten vom Leitfunktions-Wert (P502/P503). |
| **[-14] =** | **Istwert 2 Leitfunktion** (P502[-02]) | |
| **[-15] =** | **Istwert 3 Leitfunktion** (P502[-03]) | |
| **[-16] =** | **Istwert 4 Leitfunktion** (P502[-04]) | |
| **[-17] =** | **Istwert 5 Leitfunktion** (P502[-05]) | |
| **[-18] =** | **Statuswort PLC** | Statuswort über die PLC. |
| **[-19] =** | **Istwert 1 PLC** | Istwertdaten über die PLC. |
| **[-20] =** | **Istwert 2 PLC** | |
| **[-21] =** | **Istwert 3 PLC** | |
| **[-22] =** | **Istwert 4 PLC** | |
| **[-23] =** | **Istwert 5 PLC** | |

## 4.2 Prozesswerte

Alle analogen und digitalen Ein- und Ausgänge bzw. Bussoll- und Istwert können durch die PLC gelesen und weiterverarbeitet bzw. durch die PLC gesetzt (wenn Ausgangswert) werden. Der Zugriff auf die einzelnen Werte erfolgt über die hier nachfolgend aufgeführten Prozesswerte.

Für alle Ausgangswerte muss der Ausgang (z.B. Digitalausgänge oder PLC Sollwert) so programmiert werden, dass als Ereignisquelle die PLC vorgesehen ist.

Alle Prozessdaten werden von der PLC bei jedem neuen zyklischen Durchlauf am Anfang vom FU eingelesen und erst am Ende des PLC Programms in den Frequenzumrichter geschrieben!

In den nachfolgenden Tabellen sind alle Werte dargestellt, auf welche die PLC – Funktion direkt zugreifen kann. Auf alle anderen Prozesswerte muss über die Funktionsblöcke MC_ReadParameter oder MC_WriteParameter zugegriffen werden.

### 4.2.1 Ein- und Ausgänge

Hier sind alle Prozesswerte zusammengefasst, die das I/O- Interface des Frequenzumrichters beschreiben.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 0 | _0_Set_digital_output | Setzen digitaler Ausgänge | Bit 0: Mfr1<br>Bit 1: Mfr2<br>Bit 2: Dout1<br>Bit 3: Dout2<br>Bit 4: dig. Fkt. Aout<br>Bit 5: Dout3 (Din7)<br>Bit 6: Bus Statuswort Bit 8<br>Bit 7: Statuswort Bit 9<br>Bit 8: BusIO Bit1<br>Bit 9: BusIO Bit2<br>Bit 10: BusIO Bit3<br>Bit 11: BusIO Bit4<br>Bit 12: BusIO Bit5<br>Bit 13: BusIO Bit6<br>Bit 14: BusIO Bit7<br>Bit 15: BusIO Bit8 | INT | R/W |
| 1 | _1_Set_analog_output | Setzen analoger Ausgang FU | 10,0V = 100 | BYTE | R/W |
| 2 | _2_Set_external_analog_out1 | Setzen analoger Ausgang 1. IOE | 10,0V = 100 | BYTE | R/W |
| 3 | _3_Set_external_analog_out2 | Setzen analoger Ausgang 2. IOE | 10,0V = 100 | BYTE | R/W |
| 4 | _4_State_digital_output | Zustand digitale Ausgänge | Bit 0: Mfr1<br>Bit 1: Mfr2<br>Bit 2: Dout1<br>Bit 3: Dout2<br>Bit 4: dig. Fkt. Aout<br>Bit 5: Dout3 (Din7)<br>Bit 6: Bus Statuswort Bit 8<br>Bit 7: Statuswort Bit 9<br>Bit 8: BusIO Bit1<br>Bit 9: BusIO Bit2<br>Bit 10: BusIO Bit3<br>Bit 11: BusIO Bit4<br>Bit 12: BusIO Bit5<br>Bit 13: BusIO Bit6<br>Bit 14: BusIO Bit7<br>Bit 15: BusIO Bit8 | INT | R |
| 5 | _5_Digital_input | Zustand digitale Eingänge | Bit 0: DIN1<br>Bit 1: DIN2<br>Bit 2: DIN3<br>Bit 3: DIN4<br>Bit 4: DIN5<br>Bit 5: DIN6<br>Bit 6: DIN7<br>Bit 7: Digitalfunktion AIN1<br>Bit 8: Digitalfunktion AIN2 | INT | R |

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 6 | _6_Delay_digital_inputs | Zustand digitale Eingänge nach P475 | Bit 0: DIN1<br>Bit 1: DIN2<br>Bit 2: DIN3<br>Bit 3: DIN4<br>Bit 4: DIN5<br>Bit 5: DIN6<br>Bit 6: DIN7<br>Bit 7: Digitalfunktion AIN1<br>Bit 8: Digitalfunktion AIN2 | INT | R |
| 7 | _7_Analog_input1 | Wert Analogeingang 1 (AIN1) | 10,00V = 1000 | INT | R |
| 8 | _8_Analog_input2 | Wert Analogeingang 2 (AIN2) | 10,00V = 1000 | INT | R |
| 9 | _9_Analog_input3 | Wert Analogfunktion DIN2 | 10,00V = 1000 | INT | R |
| 10 | _10_Analog_input4 | Wert Analogfunktion DIN3 | 10,00V = 1000 | INT | R |
| 11 | _11_External_analog_input1 | Wert analoger Eingang 1 (1.IOE) | 10,00V = 1000 | INT | R |
| 12 | _12_External_analog_input2 | Wert analoger Eingang 2 (1.IOE) | 10,00V = 1000 | INT | R |
| 13 | _13_External_analog_input3 | Wert analoger Eingang 1 (2.IOE) | 10,00V = 1000 | INT | R |
| 14 | _14_External_analog_input4 | Wert analoger Eingang 2 (2.IOE) | 10,00V = 1000 | INT | R |
| 15 | _15_State_analog_output | Zustand analoger Ausgang | 10,0V = 100 | BYTE | R |
| 16 | _16_State_ext_analog_out1 | Zustand Analogausgang (1. IOE) | 10,00V = 1000 | INT | R |
| 17 | _17_State_ext_analog_out2 | Zustand Analogausgang (2. IOE) | 10,00V = 1000 | INT | R |

Tabelle 104 Prozesswerte Ein- und Ausgänge

### 4.2.2 Soll- und Istwerte PLC

Die hier aufgeführten Prozesswerte bilden die Schnittstelle der PLC zum Frequenzumrichter. Die Funktion der PLC Sollwerte wird im (P553) festgelegt.

**HINWEIS**

Der Prozesswert **_20_PLC_control_word** überschreibt den Funktionsblock MC_Power. Die PLC Sollwerte überschreiben die Funktionsblöcke MC_Move…. und MC_Home.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 20 | _20_PLC_control_word | PLC Steuerwort | Entspricht USS Profil | INT | R/W |
| 21 | _21_PLC_set_val1 | PLC Sollwert 1 | 100% = 4000h | INT | R/W |
| 22 | _22_PLC_set_val2 | PLC Sollwert 2 | 100% = 4000h | INT | R/W |
| 23 | _23_PLC_set_val3 | PLC Sollwert 3 | 100% = 4000h | INT | R/W |
| 24 | _24_PLC_set_val4 | PLC Sollwert 4 | 100% = 4000h | INT | R/W |
| 25 | _25_PLC_set_val5 | PLC Sollwert 5 | 100% = 4000h | INT | R/W |
| 26 | _26_PLC_additional_control_word1 | PLC Zusatzsteuerwort 1 | Entspricht USS Profil | INT | R/W |
| 27 | _27_PLC_additional_control_word2 | PLC Zusatzsteuerwort 2 | Entspricht USS Profil | INT | R/W |
| 28 | _28_PLC_status_word | PLC Statuswort | Entspricht USS Profil | INT | R/W |
| 29 | _29_PLC_act_val1 | PLC Istwert 1 | 100% = 4000h | INT | R/W |
| 30 | _30_PLC_act_val2 | PLC Istwert 2 | 100% = 4000h | INT | R/W |
| 31 | _31_PLC_act_val3 | PLC Istwert 3 | 100% = 4000h | INT | R/W |
| 32 | _32_PLC_act_val4 | PLC Istwert 4 | 100% = 4000h | INT | R/W |
| 33 | _33_PLC_act_val5 | PLC Istwert 5 | 100% = 4000h | INT | R/W |
| 34 | _34_PLC_Busmaster_Control_word | Steuerwort der Leitfunktion (Busmasterfunktion) über PLC | Entspricht USS Profil | INT | R/W |

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 35 | _35_PLC_32Bit_set_val1 | 32Bit PLC Sollwert<br>- P553[1] = Low Word des 32Bit Wert<br>- P553[2] = High Word des 32Bit Wert | _ | DINT | R/W |
| 36 | _36_PLC_32Bit_act_val1 | 32Bit PLC Istwert<br>- PLC Istwert 1 = Low Word des 32Bit Wert<br>- PLC Istwert 2 = High Word des 32Bit Wert | _ | DINT | R/W |

Tabelle 105 Prozesswerte PLC Soll- und Istwerte

### 4.2.3 Bus Soll- und Istwerte

Diese Prozesswerte spiegeln alle Soll- und Istwerte wieder, die über die verschiedenen Bussysteme in den Frequenzumrichter gelangen.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 40 | _40_Inverter_status | FU Statuswort | Entspricht USS Profil | INT | R |
| 41 | _41_Inverter_act_val1 | FU Istwert 1 | 100% = 4000h | INT | R |
| 42 | _42_Inverter_act_val2 | FU Istwert 2 | 100% = 4000h | INT | R |
| 43 | _43_Inverter_act_val3 | FU Istwert 3 | 100% = 4000h | INT | R |
| 44 | _44_Inverter_act_val4 | FU Istwert 4 | 100% = 4000h | INT | R |
| 45 | _45_Inverter_act_val5 | FU Istwert 5 | 100% = 4000h | INT | R |
| 46 | _46_Inverter_lead_val1 | Broadcast Master Funktion: Leitwert 1 | 100% = 4000h | INT | R |
| 47 | _47_Inverter_lead_val2 | Broadcast Master Funktion: Leitwert 2 | 100% = 4000h | INT | R |
| 48 | _48_Inverter_lead_val3 | Broadcast Master Funktion: Leitwert 3 | 100% = 4000h | INT | R |
| 49 | _49_Inverter_lead_val4 | Broadcast Master Funktion: Leitwert 4 | 100% = 4000h | INT | R |
| 50 | _50_Inverter_lead_val5 | Broadcast Master Funktion: Leitwert 5 | 100% = 4000h | INT | R |
| 51 | _51_Inverter_control_word | Resultierendes Steuerwort Bus | Entspricht USS Profil | ÎNT | R |
| 52 | _52_Inverter_set_val1 | Resultierender Hauptsollwert 1 Bus | 100% = 4000h | INT | R |
| 53 | _53_Inverter_set_val2 | Resultierender Hauptsollwert 2 Bus | 100% = 4000h | INT | R |
| 54 | _54_Inverter_set_val3 | Resultierender Hauptsollwert 3 Bus | 100% = 4000h | INT | R |
| 55 | _55_Inverter_set_val4 | Resultierender Hauptsollwert 4 Bus | 100% = 4000h | INT | R |
| 56 | _56_Inverter_set_val5 | Resultierender Hauptsollwert 5 Bus | 100% = 4000h | INT | R |
| 57 | _57_Broadcast_set_val1 | Broadcast Slave: Nebensollwert 1 | 100% = 4000h | INT | R |
| 58 | _58_Broadcast_set_val2 | Broadcast Slave: Nebensollwert 2 | 100% = 4000h | INT | R |
| 59 | _59_Broadcast_set_val3 | Broadcast Slave: Nebensollwert 3 | 100% = 4000h | INT | R |
| 60 | _60_Broadcast_set_val4 | Broadcast Slave: Nebensollwert 4 | 100% = 4000h | INT | R |
| 61 | _61_Broadcast_set_val5 | Broadcast Slave: Nebensollwert 5 | 100% = 4000h | INT | R |
| 62 | _62_Inverter_32Bit_set_val1 | Resultierender 32Bit Hauptsollwert 1 Bus<br>- Low Word in P546[1]<br>- High Word in P546[2] | _ | DINT | R |
| 63 | _63_Inverter_32Bit_act_val1 | FU 32Bit Istwert 1<br>- Low Word in P543[1]<br>- High Word in P543[2] | _ | DINT | R |

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 64 | _64_Inverter_32Bit_lead_val1 | 32Bit Leitwert 1<br>- Low Word in P502[1]<br>- High Word in P502[2] | _ | DINT | R |
| 65 | _65_Broadcast_32Bit_set_val1 | 32Bit Broadcast Slave Nebensollwert 1<br>- Low Word in P543[1]<br>- High Word in P543[2] | _ | DINT | R |
| 66 | _66_BusIO_input_bits | Eingehende BusI/O Daten<br>- Bit1 – 8 = Bus I/O In Bit 0 – 7<br>- Bit 9 = Merker 1<br>- Bit 10 = Merker 2<br>- Bit 11 = Bit8 vom Bus Steuerwort<br>- Bit 12 = Bit9 vom Bus Steuerwort | _ | INT | R |

Tabelle 106 Prozesswerte Bus Soll- und Istwerte

### 4.2.4 ControlBox und ParameterBox

Über die hier aufgeführten Prozesswerte kann auf die Bedienboxen zugegriffen werden. Damit ist die Realisierung einfacher HMI Anwendungen möglich.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 70 | _70_Set_controlbox_show_val | Anzeigewert für die ControlBox | Anzeigewert = Bit 29 – Bit 0<br>Kommastelle = Bit 31 – Bit30 | DINT | R/W |
| 71 | _71_Controlbox_key_state | Tastaturzustand der ControlBox | Bit 0: ON<br>Bit 1: OFF<br>Bit 2: DIR<br>Bit 3: UP<br>Bit 4: DOWN<br>Bit 5: Enter | BYTE | R |
| 72 | _72_Parameterbox_key_state | Tastaturzustand der ParameterBox | Bit 0: ON<br>Bit 1: OFF<br>Bit 2: DIR<br>Bit 3: UP<br>Bit 4: DOWN<br>Bit 5: Enter<br>Bit 6: Right<br>Bit 7: Left | BYTE | R |

Tabelle 107 Prozesswerte Control- und Parameterbox

### 4.2.5 Infoparameter

Hier sind die wichtigsten Istwerte des Frequenzumrichters aufgeführt.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 80 | _80_Current_fault | aktuelle Störungsnummer | Fehler 10.0 = 100 | BYTE | R |
| 81 | _81_Current_warning | aktuelle Warnung | Warnung 10.0 = 100 | BYTE | R |
| 82 | _82_Current_reason_FI_blocked | aktuelle Ursache für den Zustand Einschaltsperre | Problem 10.0 = 100 | BYTE | R |
| 83 | _83_Input_voltage | aktuelle Netzspannung | 100 V = 100 | INT | R |
| 84 | _84_Current_frequenz | aktuelle Frequenz | 10Hz = 100 | INT | R |
| 85 | _85_Current_set_point_frequency1 | aktuelle Sollfrequenz von der Sollwertquelle | 10Hz = 100 | INT | R |
| 86 | _86_Current_set_point_frequency2 | aktuelle Sollfrequenz Umrichter | 10Hz = 100 | INT | R |
| 87 | _87_Current_set_point_frequency3 | aktuelle Sollfrequenz nach Rampe | 10Hz = 100 | INT | R |
| 88 | _88_Current_Speed | aktuelle berechnete Drehzahl | 100rpm = 100 | INT | R |
| 89 | _89_Actual_current | aktueller Ausgangsstrom | 10.0A = 100 | INT | R |
| 90 | _90_Actual_torque_current | aktueller Momentstrom | 10.0A = 100 | INT | R |

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 91 | _91_Current_voltage | aktuelle Spannung | 100V = 100 | ÎNT | R |
| 92 | _92_Dc_link_voltage | aktuelle Zwischenkreisspannung | 100V = 100 | INT | R |
| 93 | _93_Actual_field_current | aktueller Feldstrom | 10.0A = 100 | INT | R |
| 94 | _94_Voltage_d | aktuelle Spannungskomponente d-Achse | 100V = 100 | INT | R |
| 95 | _95_Voltage_q | aktuelle Spannungskomponente q-Achse | 100V = 100 | INT | R |
| 96 | _96_Current_cos_phi | aktueller Cos(phi) | 0.80 = 80 | BYTE | R |
| 97 | _97_Torque | aktuelles Drehmoment | 100% = 100 | INT | R |
| 98 | _98_Field | aktuelles Feld | 100% = 100 | BYTE | R |
| 99 | _99_Apparent_power | aktuelle Scheinleistung | 1,00KW = 100 | INT | R |
| 100 | _100_Mechanical_power | aktuelle mechanische Leistung | 1,00KW = 100 | INT | R |
| 101 | _101_Speed_encoder | aktuelle gemessene Drehzahl | 100rpm = 100 | INT | R |
| 102 | _102_Usage_rate_motor | aktuelle Auslastung Motor (Momentanw.) | 100% = 100 | INT | R |
| 103 | _103_Usage_rate_motor_I2t | aktuelle Auslastung Motor I2t | 100% = 100 | INT | R |
| 104 | _104_Usage_rate_brake_resistor | aktuelle Auslastung Bremswiderstand | 100% = 100 | INT | R |
| 105 | _105_Head_sink_temp | aktuelle Kühlkörpertemperatur | 100°C = 100 | INT | R |
| 106 | _106_Inside_temp | aktuelle Innenraumtemperatur | 100°C = 100 | INT | R |
| 107 | _107_Motor_temp | aktuelle Motortemperatur | 100°C = 100 | INT | R |
| 141 | _141_Pos_Sensor_Inc | Position des Inkrementalgeber | 0.001 Umdrehung | DINT | R |
| 142 | _142_Pos_Sensor_Abs | Position des Absolutwertgeber | 0.001 Umdrehung | DINT | R |
| 143 | _143_Pos_Sensor_Uni | Position des Universalgeber | 0.001 Umdrehung | DINT | R |
| 144 | _144_Pos_Sensor_HTL | Position des HTL-Geber | 0.001 Umdrehung | DINT | R |
| 145 | _145_Actual_pos | Istposition | 0.001 Umdrehung | DINT | R |
| 146 | _146_Actual_ref_pos | Aktuelle Sollposition | 0.001 Umdrehung | DINT | R |
| 147 | _147_Actual_pos_diff | Positionsdifferenz zwischen Soll- und Istwert | 0.001 Umdrehung | DINT | R |

Tabelle 108 Prozesswert FU Istwerte

### 4.2.6 PLC Fehler

Über den Prozesswert ErrorFlags können aus dem PLC Programm heraus die FU Fehler E23.0 bis E24.7 gesetzt werden.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 110 | _110_ErrorFlags | Erzeugt Benutzerfehler im FU | Bit 0: E 23.0<br>Bit 1: E 23.1<br>Bit 2: E 23.2<br>Bit 3: E 23.3<br>Bit 4: E 23.4<br>Bit 5: E 23.5<br>Bit 6: E 23.6<br>Bit 7: E 23.7 | BYTE | R/W |
| 111 | _111_ErrorFlags_ext | Erzeugt Benutzerfehler im FU | Bit 0: E 24.0<br>Bit 1: E 24.1<br>Bit 2: E 24.2<br>Bit 3: E 24.3<br>Bit 4: E 24.4<br>Bit 5: E 24.5<br>Bit 6: E 24.6<br>Bit 7: E 24.7 | BYTE | R/W |

Tabelle 109 Prozesswert User Error Flags

### 4.2.7 PLC Parameter

Über diese Gruppen von Prozessdaten kann direkt auf die PLC Parameter P355, P356 und P360 zugegriffen werden.

| Index | Name | Funktion | Normierung | Typ | Zugriff |
|---|---|---|---|---|---|
| 115 | _115_PLC_P355_1 | PLC INT Parameter P355 [-01] | - | INT | R |
| 116 | _116_PLC_P355_2 | PLC INT Parameter P355 [-02] | - | INT | R |
| 117 | _117_PLC_P355_3 | PLC INT Parameter P355 [-03] | - | INT | R |
| 118 | _118_PLC_P355_4 | PLC INT Parameter P355 [-04] | - | INT | R |
| 119 | _119_PLC_P355_5 | PLC INT Parameter P355 [-05] | - | INT | R |
| 120 | _120_PLC_P355_6 | PLC INT Parameter P355 [-06] | - | INT | R |
| 121 | _121_PLC_P355_7 | PLC INT Parameter P355 [-07] | - | INT | R |
| 122 | _122_PLC_P355_8 | PLC INT Parameter P355 [-08] | - | INT | R |
| 123 | _123_PLC_P355_9 | PLC INT Parameter P355 [-09] | - | INT | R |
| 124 | _124_PLC_P355_10 | PLC INT Parameter P355 [-10] | - | INT | R |
| 125 | _125_PLC_P356_1 | PLC DINT Parameter P356 [-01] | - | DINT | R |
| 126 | _126_PLC_P356_2 | PLC DINT Parameter P356 [-02] | - | DINT | R |
| 127 | _127_PLC_P356_3 | PLC DINT Parameter P356 [-03] | - | DINT | R |
| 128 | _128_PLC_P356_4 | PLC DINT Parameter P356 [-04] | - | DINT | R |
| 129 | _129_PLC_P356_5 | PLC DINT Parameter P356 [-05] | - | DINT | R |
| | | | | | |
| 130 | _130_PLC_P360_1 | PLC Anzeige Parameter P360[-01] | - | DINT | R/W |
| 131 | _131_PLC_P360_2 | PLC Anzeige Parameter P360[-02] | - | DINT | R/W |
| 132 | _132_PLC_P360_3 | PLC Anzeige Parameter P360[-03] | - | DINT | R/W |
| 133 | _133_PLC_P360_4 | PLC Anzeige Parameter P360[-04] | - | DINT | R/W |
| 134 | _134_PLC_P360_5 | PLC Anzeige Parameter P360[-05] | - | DINT | R/W |
| | | | | | |
| 135 | _135_PLC Scope Anzeigewert 1 | PLC_Scope_Int_1 | - | INT | R/W |
| 136 | _136_PLC Scope Anzeigewert 2 | PLC_Scope_Int_2 | - | INT | R/W |
| 137 | _137_PLC Scope Anzeigewert 3 | PLC_Scope_Int_3 | - | INT | R/W |
| 138 | _138_PLC Scope Anzeigewert 4 | PLC_Scope_Int_4 | - | INT | R/W |
| 139 | _139_PLC Scope Anzeigewert 5 | PLC_Scope_Bool_1 | - | INT | R/W |
| 140 | _140_PLC Scope Anzeigewert 6 | PLC_Scope_Bool_2 | - | INT | R/W |

Tabelle 110 Umrichterparameter mit die direktem Zugriff

## 4.3 PLC Störmeldungen

Störmeldungen führen zum Abschalten des Frequenzumrichters, um einen Gerätedefekt zu verhindern. Bei PLC Störmeldungen wird die Abarbeitung der PLC gestoppt und die PLC geht in den Zustand „PLC-Error“. Bei anderen Störmeldungen läuft die PLC weiter. Nach einer Quittierung des Fehlers startet die PLC wieder automatisch.

**Beim *PLC User Fault* 23.X läuft die PLC weiter!**

| Anzeige in der SimpleBox | | Störung | Ursache |
|---|---|---|---|
| Gruppe | Detail in P700[-01] / P701 | Text in der ParameterBox | Abhilfe |
| E022 | **22.0** | **Kein PLC – Programm** | Die PLC wurde gestartet es befindet sich jedoch kein PLC Programm im FU<br>- PLC Programm in den FU laden |
| | **22.1** | **PLC – Programm ist fehlerhaft** | Die Checksummen Prüfung über das PLC Programm ergab einen Fehler.<br>- FU neu starten (Power ON) und wieder versuchen<br>- PLC Programm neu laden |
| | **22.2** | **Falsche Sprungadresse** | Programmfehler, Verhalten wie im Fehler 22.1 |
| | **22.3** | **Stack Überlauf** | Es wurden in der Laufzeit des Programms mehr als 6 Klammerebenen geöffnet<br>- Programm auf Laufzeitfehler überprüfen |
| | **22.4** | **Max. PLC Zyklen überschritten** | Die angegebene max Zykluszeit des PLC Programmes wurde überschritten<br>- Zykluszeit anpassen oder Programm überprüfen |
| | **22.5** | **Unbekannter Befehlscode** | Ein im Programm vorhandener Befehlscode kann nicht ausgeführt werden, da er unbekannt ist<br>- Programmfehler, Verhalten wie im Ferhler 22.1<br>- Version der PLC und die Version von NORDCON passen nicht zusammen |
| | **22.6** | **PLC Schreibzugriff** | Wärend eines laufenden PLC Programmes wurde der Programminhalt verändert |
| | **22.9** | **PLC Sammelfehler** | Die Fehlerursache kann nicht genau aufgelöst werden<br>- Verhalten wie im Ferhler 22.1 |
| E023 | **23.0 … 23.7** | **PLC User Fault 1 … 8** | Dieser Fehler kann durch das PLC Programm ausgelöst werden, um Probleme im Ablauf des PLC Programm nach Außen zu signalisieren. Die Auslösung erfolgt über das Beschreiben der Prozessvariable „ErrorFlags“. |
| E024 | **24.0 … 24.7** | **PLC User Fault 9 … 16** | |

Tabelle 111 PLC Fehlermeldungen

# 5 Verzeichnisse / Register

## 5.1 Abkürzungen

**AE**…… Aktuelles Ergebnis
**AIN** ..... Analog Eingang
**AOUT** ... Analog Ausgang
**AWL** ..... Anwendungsliste (siehe auch IL)
**COB-ID** . Communication Object Identifier
**DI** (DIN) Digital Eingang
**DO** (DOUT) Digital Ausgang
**E/A**..... Ein- / Ausgang
**EEPROM** Nicht flüchtiger Speicher
**EMV**..... Elektromagnetische Verträglichkeit
**FB**…….. Funktionsblock
**FU** …….. Frequenzumrichter
**HSW** ..... Hauptsollwert
**I/O** ..... In-/ Out (Eingang / Ausgang)
**IL**..... Instruction List (siehe auch AWL)
**ISD**..... Feldstrom (Stromvector- Regelung)
**LED**..... Leuchtdiode
**MC** ..... Motion Control
**NSW** ..... Nebensollwert
**P-Box**... ParameterBox
**PDO** ..... Prozess Daten Objekt
**PLC**..... SPS (Speicher Programmierbare ..... Steuerung)
**S** ..... Supervisor- Parameter, P003
**SW** ..... Software-Version, P707
**STW**..... Steuerwort
**ZSW**..... Zustandswort (Statuswort)

## 5.2 Abbildungen

Abbildung 1 Speicheraufbau ..... 9
Abbildung 2 Prozessabbild ..... 10
Abbildung 3 Erzeugung von Steuerwort & Hauptsollwerten aus der Vielzahl möglicher Quellen ..... 11
Abbildung 4 PLC Editor ..... 14
Abbildung 5 Deklarationsfenster ..... 15
Abbildung 6 NORDCON Übersichtsfenster ..... 16
Abbildung 7 Prinzipdarstellung Synchronisationsablauf Fliegenden Säge ..... 62
Abbildung 8 Sollwertverarbeitung FB PIDT1 ..... 65
Abbildung 9 Darstellung auf der ParameterBox im Visualisierungsmode ..... 66

## 5.3 Tabellen

Tabelle 1 Spezifikation der PLC ..... 9
Tabelle 2 Datentypen ..... 20
Tabelle 3 Numerische Literale ..... 20
Tabelle 4 Bitweiser Zugriff auf Variablen ..... 22
Tabelle 5 Überblick Lade- und Speicheroperationen ..... 22
Tabelle 6 LD ..... 23
Tabelle 7 LDN ..... 23
Tabelle 8 ST ..... 23
Tabelle 9 STN ..... 23
Tabelle 10 Übersicht arithmetische Operatoren ..... 24
Tabelle 11 ABS ..... 24
Tabelle 12 ADD und ADD( ..... 25
Tabelle 13 DIV und DIV( ..... 25
Tabelle 14 LIMIT ..... 26
Tabelle 15 MAX ..... 26
Tabelle 16 MIN ..... 26
Tabelle 17 MUX ..... 27
Tabelle 18 MOD und MOD( ..... 27
Tabelle 19 MUL und MUL( ..... 28
Tabelle 20 SUB und SUB( ..... 28
Tabelle 21 Überblick Erweiterte mathematische Operatoren ..... 29
Tabelle 22 EXP ..... 29
Tabelle 23 LOG ..... 29
Tabelle 24 LN ..... 30
Tabelle 25 SQR ..... 30
Tabelle 26 Trigonometrische Funktionen ..... 30
Tabelle 27 Übersicht Bit Operationen ..... 31
Tabelle 28 AND und AND( ..... 31
Tabelle 29 ANDN und ANDN( ..... 32
Tabelle 30 NOT ..... 32
Tabelle 31 OR und OR( ..... 33
Tabelle 32 ANDN und ANDN( ..... 33
Tabelle 33 ROL ..... 34
Tabelle 34 ROR ..... 34
Tabelle 35 SHL ..... 34
Tabelle 36 SHR ..... 35
Tabelle 37 S und R ..... 35
Tabelle 38 XOR oder XOR( ..... 35
Tabelle 39 XORN und XORN( ..... 36
Tabelle 40 Überblick Vergleichsoperatoren ..... 37
Tabelle 41 EQ ..... 37
Tabelle 42 GE ..... 37
Tabelle 43 GT ..... 38
Tabelle 44 LE ..... 38
Tabelle 45 LT ..... 38
Tabelle 46 NE ..... 39
Tabelle 47 Überblick Sprünge ..... 40
Tabelle 48 JMP ..... 40
Tabelle 49 JMPC ..... 40
Tabelle 50 JMPCN ..... 40
Tabelle 51 Überblick Typenkonvertierung ..... 41
Tabelle 52 BYTE_TO_BOOL ..... 41

Tabelle 53 BOOL_TO_BYTE ..... 41
Tabelle 54 INT_TO_BYTE ..... 41
Tabelle 55 BYTE_TO_INT ..... 42
Tabelle 56 DINT_TO_INT ..... 42
Tabelle 57 INT_TO_DINT ..... 42
Tabelle 58 Überblick Standartbibliothek ..... 43
Tabelle 59 CTD Abwärtszähler ..... 44
Tabelle 60 CTU Aufwärtszähler ..... 44
Tabelle 61 CTUD Auf- und Abwärtszähler ..... 45
Tabelle 62 SR Flip Flop ..... 45
Tabelle 63 RS Flip Flop ..... 46
Tabelle 64 R_TRIG und F_TRIG ..... 46
Tabelle 65 TON Einschaltverzögerung ..... 47
Tabelle 66 TOF Ausschaltverzögerung ..... 48
Tabelle 67 TP Zeitimpuls ..... 49
Tabelle 68 Einstellungen für die Motion Control FB ..... 50
Tabelle 69 Überblick Motion Control ..... 51
Tabelle 70 Beispiel Ermittlung Parameterindex des Funktionsblocks ..... 51
Tabelle 71 MC_ReadParameter ..... 52
Tabelle 72 MC_WriteParameter16 / MC_WriteParameter32 ..... 53
Tabelle 73 MC_MoveVelocity ..... 54
Tabelle 74 MC_MoveAbsolute ..... 55
Tabelle 75 MC_MoveRelative ..... 56
Tabelle 76 MC_MoveAdditive ..... 56
Tabelle 77 MC_Home ..... 57
Tabelle 78 MC_Power ..... 57
Tabelle 79 Belegung der Steuereingänge ( x = Pegel am Eingang ist unwichtig) ..... 58
Tabelle 80 MC_Control ..... 58
Tabelle 81 MC_ReadStatus ..... 58
Tabelle 82 MC:ReadActualPos ..... 59
Tabelle 83 MC_Reset ..... 59
Tabelle 84 MC_Stop ..... 59
Tabelle 85 Parametrierung für Getriebefunktion ..... 60
Tabelle 86 Überblick Elektronisches Getriebe ..... 60
Tabelle 87 FB_Gearing ..... 61
Tabelle 88 FB_FlyingSaw ..... 62
Tabelle 89 FB_FunctionCurve ..... 63
Tabelle 90 Wertebereich Regelungsparameter FB_PIDT1 ..... 64
Tabelle 91 FB_PIDT1 ..... 65
Tabelle 92 FB_STRINGToPBox ..... 67
Tabelle 93 FB_DINTToPBox ..... 68
Tabelle 94 resultierende COB-ID's ..... 69
Tabelle 95 Überblick CANopen ..... 69
Tabelle 96 FB_NMT ..... 70
Tabelle 97 FB_PDOConfig ..... 71
Tabelle 98 FB_PDOSend ..... 72
Tabelle 99 FB_PDOReceipt ..... 73
Tabelle 100 FB_Capture ..... 74
Tabelle 101 FB_WriteTrace ..... 75
Tabelle 102 FB_ReadTrace ..... 76
Tabelle 103 FB_Weigh ..... 77
Tabelle 104 Prozesswerte Ein- und Ausgänge ..... 88
Tabelle 105 Prozesswerte PLC Soll- und Istwerte ..... 89
Tabelle 106 Prozesswerte Bus Soll- und Istwerte ..... 90
Tabelle 107 Prozesswerte Control- und Parameterbox ..... 90
Tabelle 108 Prozesswert FU Istwerte ..... 91
Tabelle 109 Prozesswert User Error Flags ..... 91
Tabelle 110 Umrichterparameter mit die direktem Zugriff ..... 92
Tabelle 111 PLC Fehlermeldungen ..... 93

## 5.4 Stichwort-Verzeichnis


**A**

Akku ..... 11

Auswahl Anzeige (P001) ..... 78

**B**

Betriebsanzeigen ..... 78

Bus – Istwert (P543) ..... 84

Buszustand über PLC (P353) ..... 79

**D**

Digitalausgang

Funkt. (P434) ..... 81

**F**

FB CTD Abwärtszähler ..... 44

FB CTU Abwärtszähler ..... 44

FB CTUD Auf-/Abwärtszähler ..... 45

FB R_TRIG und F_TRIG ..... 46

FB RS Flip Flop ..... 46

FB SR Flip Flop ..... 45

FB TOF Ausschaltverzögerung ..... 48

FB TON Einschaltverzögerung ..... 47

FB TP Zeitimpuls ..... 49

FB_Capture ..... 74

FB_DINTToPBox ..... 68

FB_FlyingSaw ..... 61

FB_Gearing ..... 61

FB_PDOConfig ..... 71

FB_PDOReceipt ..... 73

FB_PDOSend ..... 72

FB_PIDT1 ..... 64

FB_ReadTrace ..... 76

FB_STRINGToPBox ..... 67

FB_Weigh ..... 77

FB_WriteTrace ..... 75

Funkt. Analogausg. (P418) ..... 80

Funkt. BusIO In Bits (P480) ..... 82

Funkt. BusIO Out Bits (P481) ..... 82

**I**

Informationsparameter ..... 85

**L**

Leitfunktion Ausgabe (P503) ..... 83

**M**

MC_Control ..... 58

MC_Home ..... 57

MC_MoveAbsolute ..... 55

MC_MoveAdditive ..... 56

MC_MoveRelative ..... 56

MC_MoveVelocity ..... 54

MC_Power ..... 57

MC_ReadActualPos ..... 59

MC_ReadParameter ..... 52

MC_ReadStatus ..... 58

MC_Reset ..... 59

MC_Stop ..... 59

MC_WriteParameter ..... 53

**N**

Niederspannungsrichtlinie ..... 2

**O**

Operator ABS ..... 24

Operator ADD und ADD( ..... 25

Operator AND und AND( ..... 31

Operator ANDN und ANDN( ..... 32

Operator BOOL_TO_BYTE ..... 41

Operator BYTE_TO_BOOL ..... 41

Operator BYTE_TO_INT ..... 42

Operator DINT_TO_INT ..... 42

Operator DIV und DIV( ..... 25

Operator EQ ..... 37

Operator EXP ..... 29

Operator GE ..... 37

Operator GT ..... 38

Operator INT_TO_BYTE ..... 41

Operator INT_TO_DINT ..... 42

Operator JMP ..... 40

Operator JMPC ..... 40

Operator JMPCN ..... 40

Operator LD ..... 23

Operator LDN ..... 23

Operator LE ..... 38

Operator LIMIT ..... 26

Operator LN ..... 30

Operator LOG ..... 29

Operator LT ..... 38

Operator MAX ..... 26

Operator MIN ..... 26

Operator MOD und MOD( ..... 27

Operator MUL und MUL( ..... 28

Operator MUX ..... 27

Operator NE ..... 39

Operator NOT ..... 32

Operator OR und OR( ..... 33

Operator ORN und ORN( ..... 33, 36

Operator ROL ..... 34

Operator ROR ..... 34

Operator S und R ..... 35

Operator SHL ..... 34

Operator SHR ..... 35

Operator SQR ..... 30

Operator ST ..... 23

Operator STN ..... 23

Operator SUB und SUB( ..... 28

Operator Winkelfunktionen ..... 30

Operator XOR und XOR( ..... 35

Operator XORN und XORN ..... 36

**P**

PLC – Anzeigewert (P360) ...79, 81
PLC - Funktionalität (P350).....78
PLC – Integer Sollwert (P355) ....79
PLC - Long – Sollwert (P356) .....79
PLC – Status (P370).....79
PLC Sollwert Auswahl (P351).....78
PLC Sollwerte (P553) .....84
PLC-Parameter.....78
Prozeßdaten Bus In (P740) .....85
Prozeßdaten Bus Out (P741).....86

**Q**

Quelle Sollwerte (P510) ..... 83
Quelle Steuerwort (P509)..... 83

**R**

Regelungsparameter..... 78

**S**

Sicherheitshinweise ..... 2
Steuerklemmen..... 80
Störmeldungen..... 93
Störungen..... 93

**W**

Wert Leitfunktion (P502)..... 83

**Z**

Zusatzparameter ..... 83

NORD DRIVESYSTEMS GROUP

**Headquarters:**

**Getriebebau NORD GmbH & Co. KG**
Rudolf-Diesel-Straße 1
22941 Bargteheide, Germany
Fon +49 (0) 4532 / 289-0
Fax +49 (0) 4532 / 289-2253
info@nord.com, www.nord.com

**Member of the NORD DRIVESYSTEMS GROUP**

NORD DRIVESYSTEMS

Mat. Nr. 607 5501 / 0813